A1 参考题解

1 算法正确性证明

1.1 辗转相除法

首先, 我们定义算法的合法输入空间和算法的specification:

- 合法输入空间: $\mathbb{N}^2$.
- specification

  当有输入数字（不妨记为a, b）至少有一个为0时, 根据CLRS Chapter 31的定义:

  - $\gcd(a, 0) = \gcd(0, a) = a$.
  - $\gcd(0, 0) = 0$.

  当a, b都不为0时:

$$
\begin{aligned}
a &= p_1^{e_1} p_2^{e_2} \dots p_r^{e_r} \\
b &= p_1^{f_1} p_2^{f_2} \dots p_r^{f_r} \\
\gcd(a, b) &= p_1^{\min(e_1, f_1)} p_2^{\min(e_2, f_2)} \dots p_r^{\min(e_2, f_2)}
\end{aligned}
$$

然后我们证明该算法正确性. 证明包括两步:

1. Partial Correctness

   由gcd的数学性质可得:

   (a) $\gcd(a, b) = \gcd(b, a)$.

   (b) $\gcd(a, 0) = a$.

   (c) $\forall a > 0. \forall b > 0. \gcd(a, b) = \gcd(b, a \bmod b)$.

   我们考察输入空间的一个partition（不同的case）:

   - $a = 0 \vee b = 0$

     此时根据规则(b)知算法返回结果正确

   - $a > 0 \wedge b > 0$

     此时根据规则(a)和规则(c)知如果$\gcd(b, a \bmod b)$返回的结果正确, 那么$\gcd(a, b)$的结果也是正确的. 此处使用数学归纳法进行证明即可.

2. Termination

   算法的终止可以根据:

   - 递归的参数严格递减
   - 存在递归基

- 自然数集的良序性

得到证明.

2 算法复杂度基础

2.1 证明

2.1.1 按定义证明

我们想要得到:

$$\forall \epsilon > 0. \exists N > 0, c > 0. (n \geq N \to 0 \leq \lg n \leq c n^{\epsilon})$$

我们可以利用$\lg n = o(n)$, 得到:

$$\forall \epsilon > 0. \exists M > 0. (n \geq M \to \lg n < \epsilon n)$$

令$N = 2^M$, 我们有$\forall n \geq N.\ \lg n \geq M$. 此时根据上式$\lg \lg n < \epsilon \lg n$, 故$\lg n < n^{\epsilon}$ (此时取$c = 1$).

2.1.2 按推论证明

推论: 当$\lim\limits_{n \to +\infty} \dfrac{f(x)}{g(x)} = 0$时, $f(x) = O(g(x))$且$f(x) = o(g(x))$.

因此此处可以先考察连续取值情况下$f(x) = \lg x$和$g(x) = x^{\epsilon}$在$x \to +\infty$时的大小情况. 不难根据L'Hospital定理得到:

$$\lim_{x \to +\infty} \frac{\lg x}{x^{\epsilon}} = 0$$

然后由连续情况推得离散情况成立即可.

2.2 判断

2, 3不正确:

2. 反例: $f(n) = n^2, g(n) = 1$.
3. 反例: $f(n) = 1, g(n) = n$.

2.3 函数复杂度排序

这道题可能是大家做到的第一道排序题（但是是对复杂度的排序, 不是对自然数的排序）. 由于这道题的函数过多, 我们可以先把函数分到以下三个类中:

- $A = \{f | \forall \epsilon > 0. f = o(n^{\epsilon})\}$.
- $B = \{f | \exists d > 0. f = \Theta(n^d)\}$.
- $C = \{f | \forall d > 0. f = \omega(n^d)\}$.

这三个类的复杂度是严格递增的, 因此, 我们只需要把这三个类中的函数排好序之后, 整体的序就排好了.

正确答案:

$$
\begin{array}{llllll}
n^{1/\lg n} & =1 & \ll \lg(\lg^* n) & \ll \lg^*(\lg n) & = \lg^* n & \ll 2^{\lg^* n} \\
\ll \ln\ln n & \ll \sqrt{\lg n} & \ll \ln n & \ll \lg^2 n & \ll 2^{2\sqrt{\lg n}} & \ll (\sqrt{2})^{\lg n} \\
\ll 2^{\lg n} & =n & \ll \lg(n!) & =n\lg n & \ll n^2 & =4^{\lg n} \\
\ll n^3 & \ll (\lg n)! & \ll (\lg n)^{\lg n} & =n^{\lg\lg n} & \ll (\frac{3}{2})^n & \ll 2^n \\
\ll n2^n & \ll e^n & \ll n! & \ll (n+1)! & \ll 2^{2^n} & \ll 2^{2^{n+1}}
\end{array}
$$

图1. 某位大佬的解答

> 下面的分析中的i和n会假设为足够大的, 故在边界方面不会很严谨, 但是对于研究渐进增长率来说是足够的了.

这里挑大家作业出错比较多的几个函数分析:

1. $\lg(\lg^* n)$, $\lg^*(\lg n)$, $\lg^* n$, $2^{\lg^* n}$

   由于我们对$\lg^*(\cdot)$的性质并不了解, 故我们可以先找些实例分析一下:

   令$\lg^{(i)} n = 1$, 我们可以得到$\lg^*(n) = i$:

   - $i = 1, n = 2$
   - $i = 2, n = 2^2 = 4$
   - $i = 3, n = 2^{2^2} = 16$
   - ...

   故$n = \underbrace{2^{2^{\cdots^{2}}}}_{i}$, 不妨记为$f(2, i)$. 此时, 我们凭直觉已经猜到这是一个增长极其缓慢的函数, 不妨先将这四个函数与集合A中其它函数先分开来分别考察. 我们通过观察可以发现:

   (a) $\lg^* n \ll 2^{\lg^* n}$.

   (b) $\lg^*(\lg n) + 1 = \lg^* n$. (注意: 此处的$=$不是同阶的意思)

   (c) 若$\lg^* n = i$, 则

      i. $\lg(\lg^* n) = \lg i$.

      ii. $\lg^*(\lg n) = i - 1$.

      故 $\lg(\lg^* n) \ll \lg^*(\lg n)$.

   根据这些发现, 我们可以确立这四个函数的复杂度关系. 然后我们比较复杂度最高的$2^{\lg^* n}$和A中其它函数中复杂度最低的, 也就是$\ln\ln n$. 由于$\ln\ln n$与$\lg\lg n$同阶, 我们不妨考察$2^{\lg^* n}$与$\lg\lg n$的关系. 我们令$\lg^* n = i$, 有:

   - 由$\lg^* n = i$有$f(2, i-1) < n \le f(2, i)$

- $\lg\lg n > f(2, i-3)$
- $2^{\lg^* n} = 2^i$

当i足够大的时候, 我们不难看出$2^i \ll f(2, i-3)$.

2. $n\lg n$, $\lg n!$

有些同学这道题将两者判断为$\lg n! \ll n\lg n$, 我猜可能是因为受到$n! \ll n^n$的影响. 实际上, 由[Stirling's approximation]可以得到:

$$\lg(n!) = n\lg n - n\lg e + O(\lg n)$$

故二者其实是同阶的.