Introduction to algorithmic Fall 2017

IFT2125-6001 TA: Maëlle Zimmermann

Demonstration 8

1

Question: Let a, b $\in$ N and d = pgcd (a, b).

1. Show that there exist s, t $\in$ Z such that its + tb = d.
2. Give an efficient algorithm to compute s, t and d from a and b. The algorithm must not calculate before calculating s and t.
3. Let a, b $\in$ N such that b> 1 and gcd (a, b) = 1. Give an effective alqorith that calculate s $\in$ Z such that its mod b = 1.

Solution:

1. Suppose without loss of generality that a $\geq$ b and show the proposition by induction on b.

   Base case: b = 0: We have 1 · a + 0 · b = a = gcd (a, b)

   Induction step: b> 0: Let w = a mod b. Note that w <b, so by induction hypothesis, there exist $s_{/}$, $t_{/}$ $\in$ Z such that $s_{/}$ b + $t_{/}$ w = pgcd (b, w). according to the proof that Euclid's algorithm works, we have pgcd (a, b) = pgcd (b, w).
   So $s_{/}$ b + $t_{/}$ w = gcd (a, b) = d.

   Let s = $t_{/}$ and t = $s_{/}$ - (a ÷ b) $t_{/}$. We are getting:

$$\begin{aligned} \text{its} + tb &= t^{/}a + (s^{/ - (a \div b)\, t /})\, b \\ &= s^{/}b + t^{/}(a - (a \div b)\, b) \\ &= s^{/}b + t^{/}(a \bmod b) \\ &= s^{/}b + t^{/}w \\ &= d. \end{aligned}$$

2. We obtain directly a recursive algorithm from the previous proof:

```
def pgcd_etendu (a, b):
    if b == 0:
        return (1, 0, a)
    else :
        (s, t, d) = pgcd_extended (b, a% b)
    return (t, s - (a // b) * t, d)
```

3. Just calculate s, t $\in$ Z such that its + tb = pgcd (a, b) thanks to the algorithm previous. We are getting:

its mod b	= (sa + tb) mod b	because tb mod b = 0
	= pgcd (a, b) mod b	def. of s, t
	= 1 mod b	by hypothesis
	= 1	because b> 1.

2

Question: Give an algorithm for whole integer multiplications whose time is faster than $n^{\log_2 3}$.

Solution: We saw in B & B chapter 7 an algorithm that allows to multiply two integers of length n in a time in O ($n^{\log_2 3}$). The idea was to replace the multiplication to be calculated by three multiplications of size reduced by half.

Following the same idea, we will give an algorithm where the integers to be multiplied are separated in three rather than in two, which allows to obtain the product to be calculated from 5 multiplications of size about n / 3 (instead of 9). We will show that the time of calculation is reduced then `has O ($n^{\log_3 5}$). We illustrate the method with an example:

Let a = 123456 and b = 135790 two integers of size n = 6. We divide into 3 each integer to obtain: s = 12, t = 34, u = 56, v = 13, w = 57, x = 90.
then

$$\alpha = sv$$
$$\beta = (s + t + u) (v + w + x)$$
$$\gamma = (s - t + u) (v - w + x)$$
$$\delta = (s + 2t + 4u) (v + 2w + 4x)$$
$$\epsilon = ux$$

we have defined 5 multiplications of integers of size one third of n, so 2 here. It is possible to prove that for our example the product ab is then expressed as:

$$ab = 10_{8}\, a' + 10_{6}\, b' + 10_{4}\, c' + 10_{2}\, d' + e'$$

o`u:

$$b' = (-3a' + 6\beta - 2\gamma - \delta + 12e') \div 6$$
$$c' = (-2a' + \beta + \gamma + 12th') \div 2$$
$$d' = (3a' - 3\beta - \gamma + \delta - 12e') \div 6.$$

On this principle we can build the following recursive algorithm:

```
def product (a, b):
    if a <100 or b <100:
        return a * b
    else :
        r = int (ceil (log10 ( max (a, b)) / 3))
        r4, r3, r2, r1 = 10 ** (4 * r), 10 ** (3 * r), 10 ** (2 * r), 10 ** (r)

        s, t, u = a // r2, (a // r1)% r1, a% r1
        v, w, x = b // r2, (b // r1)% r1, b% r1

        aa = product (s, u)
        beta = product (s + t + u, v + w + x)
        gamma = product (s - t + u, v - w + x)
        delta = product (s + 2 * t + 4 * u, v + 2 * w + 4 * x)
        ee = product (u, x)

        bb = (-3 * aa + 6 * beta - 2 * gamma - delta + 12 * ee) // 6
        cc = (-2 * aa + beta + gamma - 2 * ee) // 2
        dd = (3 * aa - 3 * beta - gamma + delta - 12 * ee) // 6

    return r4 * aa + r3 * bb + r2 * cc + r1 * dd + ee
```

To analyze the time t (m) of the algorithm as a function of the number m of decimal digits from a and b, we can make the following hypotheses:

- t (m) not decreasing
- the sum of a constant number of integers of m digits has m digits.

Moreover, we can suppose that the following operations are carried out in a time in O (m):

- the calculation of r, r1, r2, r3, r4

- the addition of two numbers of O (m) digits
- the multiplication or division of a number of O (m) digits by a constant or power of 10

We start by showing that t, u, w, x have at most $\lceil m / 3 \rceil$ digits and s, v have at most $\lceil m / 3 \rceil + 1$.

If we put $l = \log_{10} \max (a, b)$, the number of decimal digits of max (a, b) is equal to $m = \lfloor l \rfloor + 1$. The algorithm defines $r = \lceil l / 3 \rceil$. By definition of the algorithm t, u, w, x have at most r digits: indeed they are the result of an operation modulo 10_r. As s, v include the remaining figures, they have at most m - 2r figures. Gold,

Let's show that $r \leq \lceil m / 3 \rceil$:

$$\begin{aligned} l \leq \lfloor l \rfloor + 1 &\Rightarrow l / 3 \leq (\lfloor l \rfloor + 1) / 3 \\ &\Rightarrow \lceil l / 3 \rceil \leq \lceil (\lfloor l \rfloor + 1) / 3 \rceil \\ &\Rightarrow r \leq \lceil m / 3 \rceil \end{aligned}$$

Let us show that $m - 2r \leq r + 1$:

$$\begin{aligned} m - 2r \leq r + 1 &\Longleftrightarrow \lfloor l \rfloor + 1 - 2\lceil l / 3 \rceil \leq \lceil l / 3 \rceil + 1 \\ &\Longleftrightarrow \lfloor l \rfloor \leq 3\lceil l / 3 \rceil \text{ (which is true)} \end{aligned}$$

Thus each number passed as an argument in the recursive calculation of aa, beta, gamma, delta, ee also has at most $\lceil m / 3 \rceil + 1$ digits. Indeed it is the result of a constant number additions of t, u, w, x, s, v, and by hypothesis the number of digits does not increase. Of moreover, the other lines of the product function take a time in O (m). So,

$$t (m) \leq 5t (\lceil m / 3 \rceil + 1) + f (m),$$

o`uf $\in$ O (m).

Because of the +1 it is impossible to directly apply the theorems seen in class for determine the order of t. However, since t is not decreasing, we can proceed similarly to exercise 3 of demonstration 4.

More precisely one poses s (m) = t (m + 3), then:

$$\begin{aligned} s (m) &= t (m + 3) \\ &\leq 5t (\lceil (m + 3) / 3 + 1) + f (m + 3) \\ &= 5t (\lceil m / 3 \rceil + 2) + f (m + 3) \\ &\leq 5t (\lceil m / 3 \rceil + 3) + f (m + 3) \\ &= 5s (\lceil m / 3 \rceil) + f (m + 3). \end{aligned}$$

We can then apply the theorem on recurrences seen in class with $a = 5$, $b = 3$, and $f \in O(m)$. By choosing $\varepsilon = 0.1$, we obtain $s \in O(m^{\log_3 5})$. Since $t(m) \le t(m + 3) = s(m)$, we obtain $t \in O(m^{\log_3 5}) = O(m^{1.4649})$.