Lab - Fundamental algorithms — Manuel Ledo

3] i) Prove that $\max\{\lambda(a), \lambda(b)\} \leq \lambda(a+b) \leq \max\{\lambda(a), \lambda(b)\}+1 \quad \forall a,b \in \mathbb{Z}_{>0}$

ii) Prove that $\lambda(a)+\lambda(b)-1 \leq \lambda(a\cdot b) \leq \lambda(a)+\lambda(b) \quad \forall a,b \in \mathbb{Z}_{>0}$

i) To ease notation let us assume without loss of generality that $b > a$ ~~$\lambda(b) = \max\{\lambda(a), \lambda(b)\}$~~

$$\lambda(b) = \left\lfloor \frac{\log_2(b)}{64} \right\rfloor + 1 \leq \underbrace{\left\lfloor \frac{\log_2(a+b)}{64} \right\rfloor + 1}_{=\lambda(a+b)} \leq \left\lfloor \frac{\log_2(2b)}{64} \right\rfloor + 1 = \left\lfloor \frac{\log_2(b)}{64} + \frac{1}{64} \right\rfloor + 1 \leq \left(\left\lfloor \frac{\log_2(b)}{64} \right\rfloor + 1 \right) + 1 = \lambda(b)+1 \quad \square$$

ii) Notice that $\lambda(a\cdot b) = \left\lfloor \frac{\log_2(a\cdot b)}{64} \right\rfloor + 1 = \left\lfloor \frac{\log_2(a)}{64} + \frac{\log_2(b)}{64} \right\rfloor + 1$ ⊛

$$⊛ \geq \left\lfloor \frac{\log_2(a)}{64} \right\rfloor + \left\lfloor \frac{\log_2(b)}{64} \right\rfloor + 1 = \lambda(a)+\lambda(b)-1$$

$$⊛ \leq \left(\left\lfloor \frac{\log_2(a)}{64} \right\rfloor + \left\lfloor \frac{\log_2(b)}{64} \right\rfloor + 1 \right) + 1 = \lambda(a) + \lambda(b)$$

$\square$

Presented on class!

9] Let R be a ring and $k, m, n \in \mathbb{N}$. Show that the "classical" multiplication of two matrices $A \in R^{k\times m}$ and $B \in R^{m\times n}$ takes $(2m-1)kn$ arithmetic operations in R

$A \in R^{k\times m}$, $B \in R^{m\times n}$ then $C \in R^{k\times n}$. For each of the $(c_j^i)_{j=1\div n}^{i=1\div k}$ $c_j^i = A^i \cdot B_j$ where A^i is the ith-row vector of A and B_j is the jth-column vector of B, both with m dimension

Therefore in order to compute $A^i B_j$ we compute m multiplications, 1 per each component of the latter vectors and $m-1$ sums, in order to sum the m multiplications. Hence $m+(m-1) = 2m-1$ arithm. ops.

Thus, since $C \in R^{k\times n}$, we are going to repeat each of the $2m-1$ arithm ops kn times, so we are going to perform

$(2m-1)kn$ arithm. ops. in total, as we wanted to show $\square$

5] Let $n \in \mathbb{Z}$ and consider $\mathbb{Z}/n\mathbb{Z}$ the ring of congruences modulo n. Show that the classical versions "modulo n" of the classical algorithms for addition and multiplication have complexity $O(\log_2(n))$ and $O(\log_2^2 n)$ respectively

• ⊕ As we have seen in class, given two multiprecission integers, let's say $a = (-1)^s \sum_{0\leq i\leq m} a_i 2^{64i}$, $b = (-1)^s \sum_{0\leq j\leq m} b_j 2^{64j}$ both with length at most λ, assuming that the basic subroutine for the addition of 2 single precission integers takes k cycles, then addition takes $k\lambda$ machine cycles, i.e., addition of 2 multiprecission integers is $O(\lambda)$ (1)

Therefore, given $n \in \mathbb{Z}$ and considering $\mathbb{Z}/n\mathbb{Z}$ we can obviously inject and represent any element modulo n as a multiprecission integer with at most $\lambda(n)$. So, in some sense, we can inject any $a,b \in \mathbb{Z}/n\mathbb{Z}$ in $\mathbb{Z}$ and perform the defined addition which is going to give us a number belonging to $\{0,\dots,2n-2\}$ for $n > 2$, $\{0,1\}$ if $n=2$ and $\{0\}$ if $n=1$

Any element $a \in \mathbb{Z}/n\mathbb{Z}$ is going to have $\lambda(a) \leq \lambda(n) = \left\lfloor \frac{\log_2(n)}{64} \right\rfloor + 1 = O(\log_2(n))$

Therefore assuming that the machine performs modulo n operation in time $k\log_2(n)$ for $k \in \mathbb{R}$ (Note that given the resulting element between 0 and $2n-2$, it just would be necessary to perform a subtraction), then, by (1) adding two el. in $\mathbb{Z}/n\mathbb{Z}$ is $O(\log_2(n))$

• ⊙ Analogously we know that given 2 multiprecission integers a and b with length $\lambda(a)$ and $\lambda(b)$ respectively we can perform multiplication in $O(\lambda(a)\lambda(b))$. Hence, since each of our elements in $\mathbb{Z}/n\mathbb{Z}$ are going to have length at most $\lambda(n)$, we are going to perform the multiplication under the same assumptions as above in $O(\lambda(n)\lambda(n)) = O((\log_2 n)^2)$