1. Reading: K. Rosen *Discrete Mathematics and Its Applications*, 2.4

2. The main message of this lecture:

The first practical algorithms come from mathematics, more precisely, from the arithmetic: the Euclidean algorithm of finding g.c.d., addition and multiplication of integers in different bases.

A surprisingly efficient (log complexity) computational procedure of finding the greatest common divisor is provided by the classical Euclidean algorithm. We will give a description of this algorithm along with the tracing of an example $gcd(45, 111) =?$

1. Divide the larger integer by the smaller: $111 = 2 \cdot 45 + 21$
2. If the remainder is 0 then the divisor is the desired *gcd*
3. Otherwise replace the larger by the remainder and go to 1.: $(45, 111) \mapsto (21, 45)$
$45 = 21 \cdot 2 + 3$, therefore $(21, 45) \mapsto (3, 21)$, $21 = 7 \cdot 3 + 0$, thus $3 = gcd(45, 111)$.

The correctness of the Euclidean algorithm is based on the following lemma.

Lemma 9.1. $a = bq + r \quad \Rightarrow \quad gcd(a, b) = gcd(r, b)$
Proof. In fact the pairs (a, b) and (r, b) have exactly the same common divisors (therefore their greatest common divisors coincide). Indeed, $a = bq + r$ yields $r = a - bq$.

$d|a \wedge d|b \quad \Rightarrow \quad d|bq \wedge d|a - bq = r \quad \Rightarrow \quad d|r \wedge d|b$
$d|r \wedge d|b \quad \Rightarrow \quad d|bq \wedge d|bq + r = a \quad \Rightarrow \quad d|b \wedge d|a$

Theorem 9.2. *The Euclidean algorithm converges for any $a \geq b > o$ and computes $gcd(a, b)$.*
Proof. Let $a \geq b > o$. Put $r_0 = a,\ r_1 = b$. Consider the steps of the algorithm:

$$\begin{array}{ll} r_0 = r_1 \cdot q_1 + r_2, & 0 \leq r_2 < r_1 \\ r_1 = r_2 \cdot q_2 + r_3, & 0 \leq r_3 < r_2 \\ \ldots\ldots\ldots\ldots & \ldots\ldots\ldots \\ r_{n-2} = r_{n-1} \cdot q_{n-1} + r_n, & 0 \leq r_n < r_{n-1} \\ r_{n-1} = r_n \cdot q_n. & \end{array}$$

The algorithm terminates since the sequence of remainders $r_2, r_3, r_4, \ldots$ is strictly descending yet nonnegative. therefore, it reaches 0 after some finite number of steps. Then $gcd(a, b) = gcd(r_0, r_1) = gcd(r_1, r_2) = \ldots gcd(r_{n-1}, r_n) = r_n$.

Example 9.2. We all know that integers can be represented in decimal notations, e.g. $5678 = 5 \cdot 1000 + 6 \cdot 100 + 7 \cdot 10 + 8 = 5 \cdot 10^3 + 6 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0$, or in binary, e.g. $(1101)_2 = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$. There is a cute mathematical theory that allows us to produce **base b expansions** of integers for any $b > 1$.

Theorem 9.3. *Let $b > 1$. Then each positive integer n can be uniquely represented in the form $n = a_k b^k + a_{k-1} b^k + \ldots + a_i b + a_0$, where $k, a_k, a_{k-1}, \ldots, a_1, a_0 \geq 0$ and $a_i < b$ $(i = 0, 1, \ldots, k)$.*
Proof. To find an expansion of n one has to keep dividing n and then the quotients by the base b. We show both the general scheme, and an example of finding the base 7 expansion of $n = 12345$.

Division	General formula for n	Example division	Representation of $n = 12345$
$n = bq_0 + a_0$	$n = bq_0 + a_0$	$12345 = 7 \cdot 1763 + 4$	$12345 = 7 \cdot 1763 + 4$
$q_0 = bq_1 + a_1$	$= b(bq_1 + a_1) + a_0$	$1763 = 7 \cdot 251 + 6$	$= 7(7 \cdot 251 + 6) + 4$
	$= b^2 q_1 + ba_1 + a_0$		$= 7^2 \cdot 251 + 7 \cdot 6 + 4$
$q_1 = bq_2 + a_2$	$= b^2(bq_2 + a_2) + ba_1 + a_0$	$251 = 7 \cdot 35 + 6$	$= 7^2(7 \cdot 35 + 6) + 7 \cdot 6 + 4$
	$= b^3 q_2 + b^2 a_2 + ba_1 + a_0$		$= 7^3 \cdot 35 + 7^2 6 + 7 \cdot 6 + 4$
....	....	$35 = 7 \cdot 5 + 0$	$= 7^3(7 \cdot 5 + 0) + 7^2 \cdot 6 + 7 \cdot 6 + 4$
....	$= b^k q_{k-1} + \ldots ba_1 + a_0$		$= 7^4 \cdot 5 + 7^3 \cdot 0 + 7^2 \cdot 6 + 7 \cdot 6 + 4$
$q_{k-1} = b \cdot 0 + a_k$	$= b^k a_k + \ldots + ba_1 + a_0$		$= 7^4 \cdot 5 + 7^3 \cdot 0 + 7^2 \cdot 6 + 7 \cdot 6 + 4$

The algorithm terminates when a quotient $q_k = 0$ is reached. The resulting b-expansion on $n = (a_k a_{k-1} \ldots a_1 a_0)_b$ (in the sample case $12345 = (506764)_7$). The algorithm above converges since $n > q_0 > q_1 > \ldots$ and every strictly descending sequence of nonnegative integers is finite. For the proof of the uniqueness of the b-base expansion of n see the slides.

Example: **Hexadecimal expansion** – base 16. Digits: $0, 1, 2, \ldots, 9, A, B, C, D, E, F$, where $A = 10, B = 11, \ldots F = 15$. $(1A2B3C)_{16} = 1 \cdot 16^5 + 10 \cdot 16^4 + 2 \cdot 16^3 + 11 \cdot 16^2 + 3 \cdot 16 + 12 = 1715004$.

Binary addition, by example of $a = 10110$ and $b = 11011$.

carry:	1	1	1	1		
a :		1	0	1	1	0
b :		1	1	0	1	1
s :	1	1	0	0	0	1

The general formulas for computing the bits in the sum and the carry are: $s_{i+1} = a_i + b_i + c_i \pmod 2$ and $c_{i+1} = \lfloor (a_i + b_i + c_i)/2 \rfloor$. The complexity of the addition algorithm is the number of bit additions required. At each step the algorithm performs two or three additions, and n steps produce the complexity $O(n)$.

Binary multiplication of $a = (1011)_2$ and $b = (1101)_2$

a :					1	0	1	1
b :					1	1	0	1
c_0 :					1	0	1	1
c_1 :				0	0	0	0	
c_2 :			1	0	1	1		
c_3 :		1	0	1	1			
carry:	1	1	1	1				
$a \cdot b$:	1	0	0	0	1	1	1	1

The complexity of multiplication, by definition, is the total number of bit additions the binary shifts by one bit (i.e. a multiplication by 2). The standard multiplication algorithm above takes 0 bit shifts for c_0, 1 bit shift for c_1, ..., $(n-1)$ bit shifts for c_{n-1}, which brings the total number of shifts to $0 + 1 + 2 + \ldots + (n-1) = (n-1)n/2 = O(n^2)$. We also have to perform additions of n-bit integer with $(n+1)$-bit integer with ... with $(2n)$-bit integer where each of those additions takes $C \cdot n$ bit additions, which brings the total amount of bit additions to $O(n^2)$. Summary: the complexity of the standard binary multiplication algorithm above is $O(n^2)$. Surprisingly, one can do much better: the textbook displays an algorithm that uses only $O(n^{1.585})$ bit operations to multiply two n-bit numbers.

Homework assignments. (due Friday 02/16).

9A:Rosen2.4-2e; 9B:Rosen2.4-8ac; 9C:Rosen2.4-36