Zajęcia 8: (rozszerzony) algorytm Euklidesa
Jedną z operacji używanych w kryptografii jest tzw. odwracanie modulo, czyli znajdywanie x takiego, że (a * x) mod n = 1, gdzie znamy a i n. Robi się to przez tzw. rozszerzony algorytm Euklidesa.

Podstawowy algorytm Euklidesa (szkolny) służy do znajdywania największego wspólnego dzielnika przez odejmowanie większej od mniejszej liczby
Przykład: 20 i 8
20, 8 ==> 12, 8 ==> 4, 8 ==> 4, 4 ==> 0, wynik = 4, czyli NWD(20,8)=4
Zamiast odejmowania lepiej użyć dzielenia modulo:
20, 8 ==> 20mod8=4, 8 ==> 8mod4 = 0, wynik = 4

Rozszerzony algorytm Euklidesa pozwala nam znaleźć dodatkowo liczby s i t, takie, że dla danych a i b (a>b) spełnione jest równanie: s a + t b = NWD(a,b),
algorytm polega na wyliczeniu kolejnych q_i, r_i, s_i i t_i, gdzie algorytm zaczynamy liczbami:
r_1=a, r_2=b, s_1 = 1, s_2=0, t_1=0, t_2=1
Proszę zwrócić uwagę że q_1 i q_2 nie są w ogóle zdefiniowane/potrzebne, r_1 musi być większe od r_2 (bo założyliśmy że a>b) a s i t mają wartość 1 wtedy gdy r =a, bo w równaniu wyżej s jest mnożone przez a (i tak samo t=1 gdy r=b bo mnożone jest t*b)

kolejne kroki liczymy według wzorów:
$q_i=r_{i-2} / r_{i-1}$; (czyli dzielenie bez reszty)
$r_i=r_{i-2} \bmod r_{i-1} = r_{i-2} - q_i * r_{i-1}$; (druga część równania to naprawdę definicja dzielenia modulo, dokładnie to samo co się dzieje w podstawowym algorytmie Euklidesa)
$s_i = s_{i-2} - q_i * s_{i-1}$; (to wygląda jak dzielenie modulo ale q_i pochodzi z dzielenia r_i wyżej)
$t_i = t_{i-2} - q_i * t_{i-1}$ (symetrycznie jak dla s_i)
Jak robimy to na kartce to dobrze sobie umieścić wyniki w tabelce
Przykład1 20 i 8:

q_i	r_i	s_i	t_i
	20	1	0
	8	0	1
2	**4**	**1**	**-2**
2	**0**		

W momencie kiedy dostaniemy r_i=0, to wyniki dostajemy w wierszu wyżej:
$NWD(a,b)=r_{i-1}$, $s=s_{i-1}$, $t=t_{i-1}$
czyli dostaliśmy s=1, t=2, NWD(20,8)=4
sprawdzenie: 1*20 - 2 *8 = 4, wszystko się zgadza

Żeby wykonać odwracanie modulo przez rozszerzony algorytm Euklidesa, a i n muszą być względnie pierwsze, tj. NWD(a,n) = 1, czyli nasze równanie wygląda

a*s + n*t = NWD(a,n)=1 // dzielimy modulo prze n
(a*s + n*t) mod n = 1 mod n
n*t mod n = 0, bo n*t się dzieli przez n, a 1 mod n to jest zawsze 1 (jeżeli n>1), czyli dostajemy równanie od którego zaczeliśmy:
a*s mod n =1

UWAGA: 't' nam się skróciło, więc nie potrzebujemy go teraz w tabelce i liczymy tylko q_i, r_i, s_i.

Przykład2: (7 * s) mod 10 = 1, s=?

q_i	r_i	s_i

	10	0
	7	1
1	3	-1
2	**1**	**3**

W momencie kiedy dostaniemy r_i =1, to następne r_i musi być równe 0, czyli możemy przerwać algorytm na tym kroku i odpowiedź to odpowiednie s_i.
NWD(7,10)=1, s=3.
Sprawdzenie 7*3 mod 10 =21 mod 10 = 1

Przykład3: (5 * s) mod 11 = 1, s=?

q_i	r_i	s_i
	11	0
	5	1
2	**1**	**-2**

Dostaliśmy odpowiedź -2, teoretycznie spełnia ono równanie, bo 5* -2 mod 11 = -10 mod 11 = 1
(-10 to -1 * 11 + 1, dlatego reszta z dzielenia przez -1 to 1)
Ale zwykle chcemy żeby rozwiązanie naszego równania było dodatnie, możemy wtedy dodać do naszego 's' 'a', czyli w tym przypadku
s=-2 + 11 = 9 też spełnia: 5 * 9 mod 11 = 45 mod 11 = 1

Uwaga, równania 5s mod 11 = 1 i 11s mod 5 =1 będą się zaczynały tak samo w kolumnie r: r_1=11, r_2=5, ale różnią się tym czy s_1=1 (w drugim przypadku), czy s_2=1 (w pierwszym przypadku), czyli s=1 przy tej liczbie przez którą jest mnożony.

Przykład4, jeżeli np. spróbujemy rozwiązać równanie 20*x mod 8 =1 to w powyższej metodzie w r_1 nigdy nie dostaniemy 1 (w tabelce dostaniemy 4, i zaraz potem 0). I faktycznie wtedy równianie nie ma rozwiązania (proszę zwrócić uwagę, ze 20*x jest zawsze parzyste, jak podzielimy modulo przez 8 to wciąż będzie parzyste, a po lewej stronie mamy 1, liczbę nie parzystą).

Podsumowując algorytm wygląda tak:

- liczymy kolejne kroki algorytmu i sprawdzamy na każdym kroku r_i
- jeżeli r_i=1 i s_i>0, to rozwiązanie to s
- jeżeli r_i=1 i s_i<0, to rozwiązanie to s+a
- jeżeli r_i=0 (i nie mielismy na poprzednim kroku r_i=1) – równanie nie ma rozwiązań (można zwrócić np. -1)

Zadanie dla państwa:
zrobić funkcję invmod(a, n), która zwraca rozwiązanie x równania (a x) mod n =1 jeżeli są rozwiązania, a -1 jeżeli rozwiązanie nie istnieje

Proszę przysłać kod oraz rozwiązania równań (jeśli istnieją) na mój adres e-mail:
(337*x) mod 123=1
(4321*x) mod 1234=1
(432 *x) mod 321 = 1