Criptografie - Tema 2

Popescu Paul - Constantin
Facultatea de Matematică

1. Găsiți numărul minim și maxim de pași pentru algoritmul lui Euclid. Explicați proprietățile.

Cel mai favorabil caz pentru algoritmul lui Euclid este atunci când unul dintre numere este un multiplu al celuilalt. De exemplu:

$$\text{cmmdc}(26, 13)$$

in care algoritmul se opreste dupa un singur paș, returnand 13 (26 mod 13 = 0).

Cel mai nefavorabil caz pentru algoritmul lui Euclid este atunci când cele 2 numere fac parte din sirul lui Fibonacci, deoarece sunt nevoie de $n-1$ pași pentru a ajunge la un rezultat. Exemplu:

$$\text{cmmdc}(34, 21) =$$

$$\begin{aligned}
34 &= 1 \cdot 21 + 13 \\
21 &= 1 \cdot 13 + 8 \\
13 &= 1 \cdot 8 + 5 \\
8 &= 1 \cdot 5 + 3 \\
5 &= 1 \cdot 3 + 2 \\
3 &= 1 \cdot 2 + 1 \\
2 &= 1 \cdot 1 + 1
\end{aligned}$$

Proprietăți:

1. Eficienta: algoritmul are o complexitate logaritmica: $O(\log n)$, acest fiind foarte eficient.
2. Generalitate: funcționează pentru orice pereche de numere întregi pozitive.

2. Găsiți numărul de operații elementare pentru algoritmul lui Euclid.

Avand in vedere **Exercitiul 1**, algoritmul lui Euclid are o complexitate logaritmica. Deoarece fiecare paș reduce numărul mai mare la un rest mai mic decât cel anterior, rezulta ca numărul de pași este proporțional cu numărul mai mic dintre cele două. Deci putem spune ca complexitatea algoritmului lui Euclid este: $O\left(\log(\min(a, b))\right)$.

Pentru cazul cel mai nefavorabil (numerele fac parte din Sirul lui Fibonacci), descoperim o relatie interesanta cu numărul de aur $\phi \approx 1.618$ astfel:

Pentru exemplul de la **Exercitiul 1**, numărul de pași este $\log_{\phi}(21) \approx 7$, chiar numărul de pași al algoritmului.

3. Găsiți numărul de operații elementare pentru algoritmul lui Euclid extins.
Pentru varianta extinsa a algoritmului lui Euclid trebuie sa calculam suplimentar inca 2 recurente, aceastea fiind:

$$x_i = x_{i-2} - q_i x_{i+1} \quad \text{si} \quad y_i = y_{i-2} - q_i y_{i+1}$$

Deoarece aceste operatii sunt doar niste adunari si inmultiri, iar algoritmul parcurge acelasi număr de pași ca in versiunea standard, complexitatea ramane aceiasi: $O(\log n)$ (n reprezentand $\min(a, b)$, valorile initiale).

4. Demonstrati formula $\displaystyle\sum_{d|n} \varphi(n) = n$.

Demonstrație: Funcția lui Euler $\varphi(d)$ numără numerele întregi pozitive mai mici sau egale cu d care sunt prime între ele cu d. Trebuie să demonstrăm că suma valorilor $\varphi(d)$, pentru toți divizorii d ai lui n, este egală cu n.

Considerăm suma:

$$\sum_{d|n} \varphi(d).$$

Observăm că putem rescrie această sumă astfel:

$$\sum_{d|n} \varphi(d) = \sum_{d|n} \varphi\left(\frac{n}{d}\right).$$

Funcția $\varphi\left(\frac{n}{d}\right)$ numără toate numerele mai mici sau egale cu $\frac{n}{d}$ care sunt prime între ele cu $\frac{n}{d}$. Dar fiecare astfel de număr corespunde exact unei clase de resturi modulo n, unde fiecare număr între 1 și n este numărat exact o dată, având un cel mai mare divizor comun egal cu d.

Prin urmare, suma finală reprezintă exact numărul total al elementelor din $\{1, 2, \ldots, n\}$, ceea ce înseamnă:

$$\sum_{d|n} \varphi(d) = n.$$

□

5. Scrieți o variantă de cod pentru calculul funcției indicatoare a lui Euler. Asigurați-vă că funcționează și pentru numere mari. (nu apelați la librării)

`functieIndicatoareEuler.hpp`:

```
#include <iostream>
using namespace std;

long long functieIndicatoareEuler(long long numar) {
   if (numar == 1) return 1;

   long long rezultat = numar;
   long long p = 2;

   while (p * p <= numar) {
      if (numar % p == 0) {
         while (numar % p == 0)
         numar /= p;
         rezultat -= rezultat / p;
      }
      if (p == 2)  p++;
      else p += 2;
   }

   if (numar > 1)
   rezultat -= rezultat / numar;

   return rezultat;
}
```

`main.cpp`:

```
#include "functieIndicatoareEuler.hpp";

int main() {
   long long numar;
   cout << "Introdu un numar: ";
   cin >> numar;

   cout << "phi(" << numar << ") = " << functieIndicatoareEuler(numar) << endl;
   return 0;
}
```

6. Utilizați algoritmul lui Euclid extins pentru a calcula CMMDC pentru 12345 și 21100 și găsiți coeficienții Bezout.

$$\text{d) } (12345, 21100) = 5$$

$$\begin{aligned}
21100 &= 1 \cdot 12345 + 8765 & x_{21100} &= (1,0), x_{12345} = (0,1) \\
12345 &= 1 \cdot 8765 + 3580 & x_{8765} &= x_{21100} - 1 \cdot x_{12345} = (1,0) - 1 \cdot (0,1) = (1,-1) \\
8765 &= 2 \cdot 3580 + 1605 & x_{3580} &= x_{12345} - 1 \cdot x_{8765} = (0,1) - (1,-1) = (-1,2) \\
3580 &= 2 \cdot 1605 + 370 & x_{1605} &= x_{8765} - 2 \cdot x_{3580} = (1,-1) - 2 \cdot (-1,2) = (3,-5) \\
1605 &= 4 \cdot 370 + 125 & x_{370} &= x_{3580} - 2 \cdot x_{1605} = (-1,2) - 2 \cdot (3,-5) = (-7,12) \\
370 &= 2 \cdot 125 + 120 & x_{125} &= x_{1605} - 4 \cdot x_{370} = (3,-5) - 4 \cdot (-7,12) = (31,-53) \\
125 &= 1 \cdot 120 + 5 & x_{120} &= x_{370} - 2 \cdot x_{125} = (-7,12) - 2 \cdot (31,-53) = (-69,118) \\
120 &= 24 \cdot 5 + 0 & x_{5} &= x_{125} - 1 \cdot x_{120} = (31,-53) - (-69,118) = (100,-171)
\end{aligned}$$

$$\Rightarrow 5 = 100 \cdot 12345 - 171 \cdot 21100$$

7. Calculați inversul modular al lui 39 modulo 67.

$$39x \equiv 1 \pmod{67} \qquad \Rightarrow 39x + 67y = 1$$

$$\begin{aligned}
67 &= 1 \cdot 39 + 28 & x_{67} &= (1,0), x_{39} = (0,1) \\
39 &= 1 \cdot 28 + 11 & x_{28} &= x_{67} - 1 \cdot x_{39} = (1,0) - (0,1) = (1,-1) \\
28 &= 2 \cdot 11 + 6 & x_{11} &= x_{39} - 1 \cdot x_{28} = (0,1) - (1,-1) = (-1,2) \\
11 &= 1 \cdot 6 + 5 & x_{6} &= x_{28} - 2 \cdot x_{11} = (1,-1) - 2 \cdot (-1,2) = (3,-5) \\
6 &= 1 \cdot 5 + 1 & x_{5} &= x_{11} - 1 \cdot x_{6} = (-1,2) - (3,-5) = (-4,7) \\
5 &= 5 \cdot 1 + 0 & x_{1} &= x_{6} - 1 \cdot x_{5} = (3,-5) - (-4,7) = (7,-12)
\end{aligned}$$

$$\Rightarrow 1 = 7 \cdot 39 - 12 \cdot 67$$

Astfel, inversul modular al lui 39 modulo 67 este 7, deoarece:

$$39 \cdot 7 \equiv 1 \pmod{67}$$