Temă

1 Algoritmul lui Euclid prin împărțiri repetate

1.1 Enunțul problemei

Input. $a, b \in \mathbb{N}$.

Output. `da` și $\gcd(a, b)$ dacă a și b au un cel mai mare divizor comun; `nu` altfel.

Domeniu. $d = \gcd(a, b)$ dacă și numai dacă $d = \max\{d : d \in \mathbb{N}, d \mid a, d \mid b\}$.

1.2 Algoritm

```
1  gcd(a, b) {
2      if (a + b == 0)
3          return "nu";
4      while (b != 0) {
5          r = a % b;
6          a = b;
7          b = r;
8      }
9      return ["da", a];
10 }
```

1.3 Demonstrație

Cazul 1. $a = 0$ și $b = 0$. În acest caz, $a + b = 0$, așa că algoritmul intră în `if` și returnează `"nu"`, ceea ce este corect din moment ce $\gcd(a, b)$ nu există. $\square$

Cazul 2. $a > 0$ și $b = 0$. În acest caz, cum $b = 0$, nu se mai intră în `while`, ci se returnează direct a, ceea ce este corect din moment ce $\gcd(a, b) = \gcd(a, 0) = a$. $\square$

Cazul 3. $a > 0$ și $b > 0$. În acest caz, algoritmul intră în `while`. Dacă notăm cu a_i și b_i valorile variabilelor a și b imediat după a i-a iterație a buclei `while`, obținem șirul

$$(a_0, b_0) \to (a_1, b_1) \to \cdots \to (a_k, b_k).$$

Mai întâi, vom arăta că `while`-ul păstrează următorul invariant:

$$\gcd(a_i, b_i) = \gcd(a_{i+1}, b_{i+1}), \forall i = \overline{0, k-1}.$$

Cum liniile 5-7 duc fiecare pereche (a_i, b_i) în $(b_i, a_i \bmod b_i)$, avem că

$$\begin{aligned} &\gcd(a_i, b_i) = \gcd(a_{i+1}, b_{i+1}) \\ \Leftrightarrow &\gcd(a_i, b_i) = \gcd(b_i, \underbrace{a_i \bmod b_i}_{r}). \end{aligned}$$

$\exists!\, q \in \mathbb{N} : a_i = qb_i + r$. Fie d un divizor oarecare al lui b_i. Avem că $d \mid qb_i + r \leftrightarrow d \mid r$. Prin urmare, perechile (a_i, b_i) și (b_i, r) au aceeași mulțime de divizori comuni, de unde $\gcd(a_i, b_i) = \gcd(b_i, r) = \gcd(a_{i+1}, b_{i+1})$. Acum putem afirma că ultima pereche din șirul iterațiilor, $(a_k, 0)$, are gcd-ul căutat, iar acesta este egal cu valoarea returnată, a_k. $\square$

Mai rămâne să demonstrăm că `while`-ul se termină întotdeauna într-un număr finit de pași. Altfel spus, că șirul de mai sus este finit. Acest lucru este adevărat, pentru că toate valorile a_i și b_i sunt pozitive, iar la fiecare iterație unul dintre cele două numere devine mai mic decât era inițial ($a_i \to a_i \bmod b_i < a_i$ când $a_i \geq b_i$). Excepție face cazul în care $a_i < b_i$, dar acesta poate apărea doar înainte de prima iterație, și se rezolvă imediat, deoarece $(a_1, b_1) = (b_0, a_0)$. $\square$

1.4 Complexitate

Fie următoarele trei iterații consecutive ale algoritmului:

$$(a, b) \to (b, c) \to (c, d).$$

Vom presupune că $a > b$, de unde și $b > c$ și $c > d$. Știm că $d = b \bmod c \Rightarrow \exists!\, q \in \mathbb{N} : b = qc + d$. Dar $b > c > d$, așa că $q > 0$. Prin urmare,

$$b \geq c + d.$$

Dar $a > b$, deci mai avem că

$$a > c + d.$$

Adunând cele două relații, obținem că

$$a + b > 2(c + d).$$

Așadar, la fiecare două iterații, suma valorilor a și b devine de cel puțin două ori mai mică. Asta înseamnă că numărul maxim de iterații din `while` este aproximativ $2\lfloor \log_2(a + b) \rfloor$. În concluzie, complexitatea algoritmului este de ordinul $O(\log(a + b))$. $\square$

1.5 Exercițiul 3

Problema de la Exercițiul 1 nu este aceeași cu problema de la Exercițiul 3, deoarece input-urile diferă. La a doua problemă putem avea $a = b = 0$, pe când la prima nu.

2 Sortarea prin selecție

2.1 Enunțul problemei (Exercițiul 6)

Input. $n \in \mathbb{N}$ și un vector $v = \langle v_0, v_1, \ldots, v_{n-1} \rangle$, unde $v_i \in \mathbb{N}$, $\forall i = \overline{0, n-1}$.

Output. $w = \langle w_0, w_1, \ldots, w_{n-1} \rangle$, unde $w_0 \leq w_1 \leq \cdots \leq w_{n-1}$ și w este o permutare a lui v.

Domeniu. w este o permutare a lui v dacă și numai dacă, pentru orice $x \in \mathbb{N}$, numărul de apariții ale lui x în w este egal cu numărul de apariții lui x în v.

2.2 Algoritm (Exercițiul 7)

```
1   sort(v) {
2       n = v.size();
3       for (i = 0; i < n; i++) {
4           k = i;
5           for (j = i + 1; j < n; j++)
6               if (v[j] < v[k])
7                   k = j;
8           temp = v[i];
9           v[i] = v[k];
10          v[k] = temp;
11      }
12      return v;
13  }
```

2.3 Demonstrație

Remarcăm următorul invariant: La începutul celei de-a i-a iterații din primul `for`, cele mai mici i elemente din vectorul v se află la începutul său, sortate. Altfel spus, $v[j]$ este al $(j + 1)$-lea cel mai mic element din v, $\forall j < i$.

Vom demonstra prin inducție că invariantul este adevărat: Inițial, acesta e satisfăcut în mod trivial. Apoi, presupunând că invariantul este adevărat după a $(i - 1)$-a iterație, arătăm că rămâne adevărat și după a i-a:

Liniile 4-7 calculează în k poziția celui mai mic element din secvența $v[i, n)$. Cum toate elementele din $v[i, n)$ sunt mai mari sau egale cu orice element din $v[0, i)$, înseamnă că $v[k]$ va fi al $(i + 1)$-lea cel mai mic element din v. Liniile 8-10 interschimbă elementele $v[i]$ și $v[k]$, mutând într-adevăr al $(i + 1)$-lea cel mai mic element pe poziția i. $\square$

2.4 Complexitate

Linie	Cost operație	Repetări
2	1	1
3	$2n$	1
4	1	$i = \overline{0, n-1}$
5	$2(n-i-1)$	$i = \overline{0, n-1}$
6	1	$i = \overline{0, n-1}, j = \overline{i+1, n-1}$
7	$\tau_{ij} \in \{0, 1\}$	$i = \overline{0, n-1}, j = \overline{i+1, n-1}$
8	1	$i = \overline{0, n-1}$
9	1	$i = \overline{0, n-1}$
10	1	$i = \overline{0, n-1}$

Însumând costurile operațiilor elementare, obținem:

$$\begin{aligned}
T(n) &= 1 + 2n + \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{n-1} 2(n-i-1) + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} 1 + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} \tau_{ij} + \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{n-1} 1 \\
&= 1 + 2n + n + 3\sum_{i=0}^{n-1}(n-i-1) + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} \tau_{ij} + n + n + n \\
&= 1 + 6n + 3\left(n^2 - \sum_{i=0}^{n-1} i - n\right) + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} \tau_{ij} \\
&= 1 + 6n + 3\left(n^2 - \frac{(n-1)n}{2} - n\right) + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} \tau_{ij} \\
&= 1 + 6n + 3\left(\frac{n^2}{2} - \frac{n}{2}\right) + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} \tau_{ij} \\
&= 1 + \frac{9}{2}n + \frac{3}{2}n^2 + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} \tau_{ij}.
\end{aligned}$$

Dar $\tau_{ij} \in \{0, 1\}$, deci $\underbrace{1 + (9/2)n + (3/2)n^2}_{\Omega(n^2)} \leq T(n) \leq \underbrace{1 + 4n + 2n^2}_{O(n^2)}$. Așadar, $T(n) = \Theta(n^2)$. □