

АНАЛИЗ НА АЛГОРИТМИ — КОРЕКТНОСТ И СЛОЖНОСТ  
КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ” — СУ, ФМИ  
(ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 1. ПОТОК; 14 МАРТ 2018 Г.)

ВАРИАНТ № 1

Разглеждаме два алгоритъма за разпознаване на прости числа — `isPrime1` и `isPrime2`.  
Входът и на двата алгоритъма е цяло положително число  $n \geq 2$ .

```
isPrime1(n) // n ≥ 2
for k ← 2 to n-1 do
  if n се дели на k
    return false
return true

isPrime2(n) // n ≥ 2
return not hasDivisor(n, 2, n-1)

hasDivisor(n, k, r)
// проверява дали n има делител
// между k и r включително (k ≤ r)
if k = r
  if n се дели на k
    return true
  else
    return false
if k + 7 > r
  for j = k to r do
    if n се дели на j
      return true
  return false
for j = 1 to 6 do
  if hasDivisor(n, k+(j-1)⌊ $\frac{r-k}{7}$ ⌋, k+j⌊ $\frac{r-k}{7}$ ⌋)
    return true
return hasDivisor(n, k+6⌊ $\frac{r-k}{7}$ ⌋, r)
```

**Задача 1.** Докажете, че `isPrime1` е коректен алгоритъм,  
като формулирате и докажете подходяща инварианта на цикъла. (15 точки)

**Задача 2.** Докажете по индукция, че `hasDivisor` е коректен алгоритъм. (15 точки)  
(От това следва, че и `isPrime2` е коректен алгоритъм.)  
Отбележете изрично типа и параметъра на индукцията.

**Задача 3.** Намерете сложността по време на алгоритъма `isPrime2`. (10 точки)

**Задача 4.** Задачата за търговския пътник може да се реши с различни алгоритми:  
— Пълно изчерпване. Неговата времева сложност е  $\Theta(n!)$ , където  $n$  е броят на градовете.  
— Алгоритъм на Хелд—Карп. Времевата сложност на този алгоритъм е  $\Theta(n^2 \cdot 2^n)$ .  
Кой от двата алгоритъма е по-бърз? Отговорът да се обоснове подробно! (10 точки)

**АНАЛИЗ НА АЛГОРИТМИ — КОРЕКТНОСТ И СЛОЖНОСТ**  
**КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ” — СУ, ФМИ**  
**(ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 1. ПОТОК; 14 МАРТ 2018 Г.)**

**ВАРИАНТ № 2**

Разглеждаме два алгоритъма за събиране на редица от  $n$  цели числа,  $n \geq 1$

<pre> ALG-1(A[1...n] : integers) i ← 1, j ← 2, s ← 0 while i &lt; n do     s ← s + A[i] - 2A[j]     i ← i + 1     j ← j + 1 s ← s - 2A[1] + A[n] return -s         </pre>	<pre> ALG-2 (A [1...n] ) // n ≥ 1 return sum (A [1...n] )  sum (A [k...r] ) // пресмята сбора на елементите с индекси // между k и r включително (k ≤ r) if k = r     return A [k] if k + 3 &gt; r     s ← 0     for j = k to r do         s ← s + A [j]     return s s1 ← sum (A [k...k + ⌊ (r-k) / 3 ⌋ ]) s2 ← sum (A [k + ⌊ (r-k) / 3 ⌋ + 1...k + 2 ⌊ (r-k) / 3 ⌋ ]) s3 ← sum (A [k + 2 ⌊ (r-k) / 3 ⌋ + 1...r ]) return s1 + s2 + s3         </pre>
---	--

**Задача 1.** Докажете, че ALG-1 е коректен алгоритъм, като формулирате и докажете подходяща инварианта на цикъла. **(15 точки)**

**Задача 2.** Докажете по индукция, че sum е коректен алгоритъм. **(15 точки)**  
 (От това следва, че и ALG-2 е коректен алгоритъм.)  
 Отбележете изрично типа и параметъра на индукцията.

**Задача 3.** Намерете сложността по време на алгоритъма ALG-2. **(10 точки)**

**Задача 4.** Детерминантата на числова матрица с  $n$  реда и  $n$  стълба може да се пресмята по различни начини:  
 — По формулата от определението за детерминанта. Това изисква събиране на произведения по всички пермутации на стълбове, затова сложността му е  $\Theta((n+1)!)$ .  
 — Чрез привеждане в триъгълен вид. Времевата сложност на този метод е  $\Theta(n^3)$ .

Кой от двата начина е по-бърз? Отговорът да се обоснове подробно! **(10 точки)**

АНАЛИЗ НА АЛГОРИТМИ — КОРЕКТНОСТ И СЛОЖНОСТ  
 КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ” — СУ, ФМИ  
 (ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 1. ПОТОК; 14 МАРТ 2018 Г.)

ВАРИАНТ № 3

```

НОД (a,b)
while a ≠ b do
    while a > b do
        a ← a - b
    while b > a do
        b ← b - a
return a

sumMatrices (A,B,C: matrices n × n) // n ≥ 1
return sumM (A,B,C,1,n,1,n) // C ← A + B

sumM (A,B,C,k,r,p,q) // k ≤ r, p ≤ q
// C[i,j] ← A[i,j] + B[i,j],
// където i се изменя от k до r включително,
// а j се изменя от p до q включително
if (k > r) or (p > q)
    return // do nothing: empty submatrix
else if (k = r) and (p = q)
    C[k,p] ← A[k,p] + B[k,p]
else
    sumM(A, B, C, k, ⌊(k+r)/2⌋, p, ⌊(p+q)/2⌋)
    sumM(A, B, C, k, ⌊(k+r)/2⌋, ⌊(p+q)/2⌋+1, q)
    sumM(A, B, C, ⌊(k+r)/2⌋+1, r, p, ⌊(p+q)/2⌋)
    sumM(A, B, C, ⌊(k+r)/2⌋+1, r, ⌊(p+q)/2⌋+1, q)
    
```

**Задача 1.** Докажете с подходяща инварианта, че алгоритъмът НОД пресмята най-големия общ делител на целите положителни числа  $a$  и  $b$ . (15 точки)

**Задача 2.** Докажете по индукция, че  $\text{sumM}$  записва в матрицата  $C[k\dots r, p\dots q]$  сбора на числовите матрици  $A[k\dots r, p\dots q]$  и  $B[k\dots r, p\dots q]$ .

(От това следва, че  $\text{sumMatrices}$  пресмята сбора на квадратни числови матрици с  $n$  реда и  $n$  стълба.)

Отбележете изрично типа и параметъра на индукцията. (15 точки)

**Задача 3.** Намерете сложността по време на алгоритъма  $\text{sumMatrices}$ . (10 точки)

**Задача 4.** Два алгоритъма имат времеви сложности  $\Theta(n^n)$  и  $\Theta(5^{n^3})$ .

Кой от двата алгоритъма е по-бърз? Отговорът да се обоснове подробно! (10 точки)

АНАЛИЗ НА АЛГОРИТМИ — КОРЕКТНОСТ И СЛОЖНОСТ  
КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ” — СУ, ФМИ  
(ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 1. ПОТОК; 14 МАРТ 2018 Г.)

ВАРИАНТ № 4

Разглеждаме два алгоритъма, които пресмятат произведението на целите числа от  $a$  до  $b$  включително, където  $a \leq b$  са две цели числа.

<pre>product1(a,b) p ← 1 for k ← a to b do   p ← p × k return p</pre>	<pre>product2(a,b) if a = b   return a else   q ← product2(a, ⌊(a+b)/2⌋)   r ← product2(⌊(a+b)/2⌋+1, b)   return q × r</pre>
---	--

**Задача 1.** С подходяща инварианта докажете коректността на product1. (15 точки)

**Задача 2.** Чрез индукция докажете коректността на product2.  
Отбележете изрично типа и параметъра на индукцията. (15 точки)

**Задача 3.** Намерете сложността по време на алгоритъма product2. (10 точки)

**Задача 4.** Два алгоритъма имат времеви сложности  $\Theta((n^n)!)$  и  $\Theta((n!)^n)$ .  
Кой от двата алгоритъма е по-бърз? Отговорът да се обоснове подробно! (10 точки)

## КРАТКИ РЕШЕНИЯ

### ВАРИАНТ № 1

**Задача 1.** Инварианта на цикъла на `isPrime1`:

$n$  не се дели на никое от числата  $2, 3, \dots, k-1$ .

**Задача 2** се решава със силна индукция по  $r-k$ .

**Задача 3.** Времевата сложност  $T(n)$  на алгоритъма `isPrime2` удовлетворява уравнението

$T(n) = 7T\left(\frac{n}{7}\right) + 1$ , чието решение се получава от мастър-теоремата:  $T(n) = \Theta(n)$ .

**Задача 4** се решава чрез логаритмуване:

$$\log(n!) = \Theta(n \log n); \quad \log(n^2 \cdot 2^n) = 2 \log n + n \log 2 = \Theta(n).$$

С помощта на граници се доказва, че

$$n = o(n \log n), \quad \text{тоест } \log(n^2 \cdot 2^n) = o(\log(n!)).$$

Понеже  $n \log n \rightarrow \infty$  при  $n \rightarrow \infty$ , то следва, че

$$n^2 \cdot 2^n = o(n!).$$

Извод: Алгоритъмът на Хелд—Карп е по-бърз от пълното изчерпване.

---

### ВАРИАНТ № 2

**Задача 1.** Инварианта на цикъла на `ALG-1`:

$$s = A[1] - (A[2] + A[3] + \dots + A[i]) - A[i] \quad \text{и } j = i + 1.$$

**Задача 2** се решава със силна индукция по  $r-k$ .

**Задача 3.** Времевата сложност  $T(n)$  на алгоритъма `ALG-2` удовлетворява уравнението

$T(n) = 3T\left(\frac{n}{3}\right) + 1$ , чието решение се получава от мастър-теоремата:  $T(n) = \Theta(n)$ .

**Задача 4** се решава чрез логаритмуване:

$$\log((n+1)!) = \Theta((n+1) \log(n+1)) = \Theta(n \log n); \quad \log(n^3) = 3 \log n = \Theta(n).$$

С помощта на граници се доказва, че

$$n = o(n \log n), \quad \text{тоест } \log(n^3) = o(\log((n+1)!)).$$

Понеже  $n \log n \rightarrow \infty$  при  $n \rightarrow \infty$ , то следва, че

$$n^3 = o((n+1)!).$$

Извод: Привеждането в триъгълен вид е по-бързо от определението.

### ВАРИАНТ № 3

**Задача 1.** Инварианта на цикъла на НОД:

Най-големият общ делител на  $a$  и  $b$  е равен на най-големия общ делител на  $A$  и  $B$ , където  $A$  и  $B$  са началните стойности на  $a$  и  $b$  съответно.

**Задача 2** се решава със силна индукция по  $(r - k) + (q - p)$ .

**Задача 3.** Времевата сложност  $T(n)$  на `sumMatrices` удовлетворява уравнението

$$T(n) = 4T\left(\frac{n}{2}\right) + 1, \text{ чието решение се получава от мастър-теоремата: } T(n) = \Theta(n^2).$$

**Задача 4** се решава чрез логаритмуване:

$$\log(n^n) = n \log n; \quad \log(5^{n^3}) = n^3 \cdot \log 5 = \Theta(n^3).$$

С помощта на граници се доказва, че

$$n \log n = o(n^3), \text{ тоест } \log(n^n) = o\left(\log(5^{n^3})\right).$$

Понеже  $n^3 \rightarrow \infty$  при  $n \rightarrow \infty$ , то следва, че

$$n^n = o(5^{n^3}).$$

Извод: По-бърз е алгоритъмът със сложност  $\Theta(n^n)$ .

---

### ВАРИАНТ № 4

**Задача 1.** Инварианта на цикъла на `product1`:

$$p = a(a+1)(a+2)(a+3)\dots(k-1).$$

**Задача 2** се решава със силна индукция по  $b - a$ .

**Задача 3.** Времевата сложност  $T(n)$  на `product2` удовлетворява уравнението

$$T(n) = 2T\left(\frac{n}{2}\right) + 1, \text{ където } n = b - a. \text{ От мастър-теоремата следва, че } T(n) = \Theta(n).$$

**Задача 4** се решава чрез логаритмуване:

$$\log((n^n)!) = \Theta(n^n \log(n^n)) = \Theta(n^{n+1} \log n); \quad \log((n!)^n) = n \log(n!) = \Theta(n^2 \log n).$$

С помощта на граници се доказва, че  $n^2 = o(n^{n+1})$ .

$$\text{Умножаваме по } \log n: \quad n^2 \log n = o(n^{n+1} \log n), \text{ тоест } \log((n!)^n) = o(\log((n^n)!)).$$

Понеже  $(n^n)! \rightarrow \infty$  при  $n \rightarrow \infty$ , то следва, че  $(n!)^n = o((n^n)!)$ .

Извод: По-бърз е алгоритъмът със сложност  $\Theta((n^n)!)$ .