

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ”
ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 2. КУРС, 2. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 1

Задача 1. Какво връща алгоритъмът ALG_1?
 Обосновете се с инварианта.

```
ALG_1(A[1...n])
s ← 0
for k ← 2 to n do
    p ← A[k] × A[k-1]
    if p < 0
        s ← s + 1
return s
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 200
    return
for i ← 1 to n do
    for j ← 1 to n do
        print i × A[j]
p ← n / 2
for k ← 1 to 64 do
    ALG_2(A[k...p])
p ← p + 1
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
print A[n]
k ← 2
while k < n do
    ALG_3(A[1...k])
    k ← k + 2
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ”
ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 2. КУРС, 2. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 2

Задача 1. Какво връща алгоритъмът ALG_1?
 Обосновете се с инварианта.

```
ALG_1(A[1...n])
s ← A[n]
for k ← 2 to n do
    s ← s + A[k-1]
return s
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 111
    return
print A[n]
for i ← 2 to n do
    for j ← i to n do
        ALG_2(A[i...j])
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
if n < 501
    return
for i ← 1 to n do
    print A[i] + 201
p ← n / 17
for k ← 1 to 17 do
    ALG_3(A[k...p])
p ← p + 1
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ”
ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 2. КУРС, 2. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 3

Задача 1. Какво връща алгоритъмът ALG_1?
 Обосновете се с инварианта.

```
ALG_1(A[1...n])
s ← 2 × A[1]
for k ← 2 to n do
    s ← s + 2 × A[k]
    s ← s - A[k-1]
return s - A[n]
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n > 304
    i ← 1
    j ← 1
    while i ≤ n do
        j ← j + 1
        print j × A[i]
        if j > n
            j ← 1
            i ← i + 1
    ALG_2(A[1...n/3])
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
print A[n]
for k ← 1 to 4 do
    ALG_3(A[1...n-1])
    ALG_3(A[1...n-2])
    ALG_3(A[2...n-1])
    ALG_3(A[3...n])
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО “ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ”
ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ”, 2. КУРС, 2. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 4

Задача 1. Какво връща алгоритъмът ALG_1?
 Обосновете се с инварианта.

```
ALG_1(a: цяло неотрицателно число)
s ← 0
while a > 0 do
    s ← s + 5
    a ← a - 1
return s
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 555
    return
print A[n]
p ← n - 1
for k ← 1 to n do
    if A[k] > 0
        ALG_2(A[2...n])
    else
        ALG_2(A[1...p])
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
if n < 190
    return
for i ← 1 to n do
    for j ← 1 to n do
        print i × j
p ← n / 3
for k ← 1 to 81 do
    ALG_3(A[k...p])
    p ← p + 1
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

РЕШЕНИЯ

ВАРИАНТ № 1

Задача 1.

Алгоритъмът ALG_1 връща броя на смените на знаците на елементите.

Под смяна на знак се разбира съседството на положително и отрицателно число.

Инварианта:

s = броя на смените на знаци в подмасива $A[1 \dots k-1]$.

Задача 2.

Рекурентно уравнение:
 $T(n) = 64T(n/2) + \Theta(n^2)$.

От първия случай на мастър-теоремата намираме решението:
 $T(n) = \Theta(n^6)$.

Задача 3.

Рекурентно уравнение:

$$T(n) = c + \sum_{k=1}^{\lfloor (n-1)/2 \rfloor} T(2k).$$

Това уравнение е с променлива дължина на историята.

Ако n е нечетно, $T(n) = T(n+1)$.

Затоа нека n е четно.

Тогава заместваме n със $n+2$ и изваждаме двете уравнения.

Получава се уравнението

$$T(n+2) = 2T(n).$$

С характеристично уравнение или чрез развиване намираме

$$T(n) = \Theta\left(\left(\sqrt{2}\right)^n\right).$$

Задача 4. Алгоритъмът ALG_2 е по-бърз от ALG_3. Сравняването на сложностите им се извършва най-лесно чрез логаритмуване.

ВАРИАНТ № 3

Задача 1.

Алгоритъмът ALG_1 връща сбора на елементите на дадения числов масив.

Инварианта:

$$s = 2A[k-1] + \sum_{m=1}^{k-2} A[m].$$

Задача 2.

Рекурентно уравнение:
 $T(n) = T(n/3) + \Theta(n^2)$.

От третия случай на мастър-теоремата намираме решението:
 $T(n) = \Theta(n^2)$.

Задача 3.

Рекурентно уравнение:

$$T(n) = 4T(n-1) + 12T(n-2) + \Theta(1).$$

С характеристично уравнение намираме решението:

$$T(n) = \Theta(6^n).$$

Задача 4. Алгоритъмът ALG_2 е по-бърз от ALG_3. Сравняването на сложностите им се извършва най-лесно чрез логаритмуване.

ВАРИАНТ № 2

Задача 1.

Алгоритъмът ALG_1 връща сбора от всички елементи на дадения числов масив.

Инварианта:

$$s = A[n] + \sum_{m=1}^{k-2} A[m].$$

Задача 2.

Рекурентно уравнение:

$$T(n) = c + \sum_{k=1}^{n-1} (n-k) T(k).$$

Уравнението е с променлива дължина на историята. Замества n със $n+1$ и изваждаме двете уравнения. Получава се уравнението

$$T(n+1) - T(n) = \sum_{k=1}^n T(k).$$

Пак замества n със $n+1$ и изваждаме последните две уравнения. Получава се $T(n+2) = 3T(n+1) - T(n)$.

С характеристично уравнение намираме решението:

$$T(n) = \Theta\left(\left(\frac{3+\sqrt{5}}{2}\right)^n\right).$$

Задача 3.

Рекурентно уравнение:

$$T(n) = 17T(n/17) + \Theta(n).$$

От втория случай на мастър-теоремата намираме решението: $T(n) = \Theta(n \log n)$.

Задача 4. Алгоритъмът ALG_3 е по-бърз от ALG_2. Сравняването на сложностите им се извършва най-лесно чрез логаритмуване.

ВАРИАНТ № 4

Задача 1.

Алгоритъмът ALG_1 връща $5A$, където A е началната стойност на параметъра a .

Инварианта:

$$s + 5a = 5A.$$

Задача 2.

Рекурентно уравнение:

$$T(n) = n \cdot T(n-1).$$

Решаваме го чрез развиване:

$$T(n) = \Theta(n!).$$

Рекурентното уравнение съдържа променлив коефициент, затова не може да бъде решено с характеристично уравнение.

Задача 3.

Рекурентно уравнение:

$$T(n) = 81T(n/3) + \Theta(n^2).$$

От първия случай на мастър-теоремата намираме решението: $T(n) = \Theta(n^4)$.

Задача 4. Алгоритъмът ALG_3 е по-бърз от ALG_2. Сравняването на сложностите им се извършва най-лесно чрез логаритмуване.

СХЕМА ЗА ТОЧКУВАНЕ

Всяка задача носи по 5 точки, а цялото контролно — най-много 20 точки.

Задача 1. Дава се по една точка за всяка от следните стъпки:

- формулиране на вярна и използваема инварианта;
- доказателство на базата на инвариантата;
- индуктивна стъпка (поддръжка на инвариантата);
- доказателство, че алгоритъмът ще завърши;
- извод за върнатата стойност.

Ако не е формулирана вярна и използваема инварианта, не се дават точки по тази задача.

Задача 2 и задача 3. Петте точки се разпределят по следния начин:

- за съставяне на рекурентно уравнение: 2 точки;
- за решаване на рекурентното уравнение: 3 точки.

Ако уравнението е грешно съставено, не се дават точки, независимо дали и как е решено.

Задача 4. Ако асимптотичното сравнение се прави чрез граници, се дават 4 точки за пресмятане на границата и 1 точка за отговора.

Ако асимптотичното сравнение се извършва чрез логаритмуване, се дават 2 точки за логаритмуването, 2 точки за границата и 1 точка за отговора.

Тези правила се прилагат, ако е избран най-рационалният начин за решаване на задачата. В противен случай се отнема 1 точка.

Тъй като отговорът може да бъде предположен с помощта на асимптотичния ред, то при грешен отговор не се дават никакви точки за другите етапи от решението.

Ако отговорът е верен, но недостатъчно обоснован, не се дават точки за отговора, а само за преодолените етапи от обосновката.

Точки по задача 4 се дават само ако са правилно пресметнати времевите сложности на алгоритмите `ALG_2` и `ALG_3`. За сравняване на една или две погрешни сложности не се дават точки.