Зад. 1. Намерете времевата сложност на алгоритъма. **(10 точки)**

```
   f(A[1...n]: array of integers; lower, upper: integers): integer
if lower = upper
   return  A[lower]
else if upper = lower + 1
   return  A[lower] × A[upper]
else
   p ← ⌊(upper + 2 × lower) / 3⌋
   q ← ⌊(lower + 2 × upper) / 3⌋
   return  f(A,lower,p) × f(A,p+1,q) × f(A,q+1,upper)
```

Решение: Сложността удовлетворява рекурентното уравнение $\mathrm{T}(n) = 3\mathrm{T}\left(\frac{n}{3}\right) + \Theta(1)$.

Първото събираемо е времето за трите рекурсивни извиквания, а второто събираемо дава времето за всички други инструкции (нерекурсивната част). От мастър-теоремата намираме времевата сложност на алгоритъма: $\mathrm{T}(n) = \Theta(n)$.

Зад. 2. Имате n празни кухненски съда с вместимости $A[1]$, $A[2]$, ... , $A[n]$ литра. Напълнете догоре максимален брой от тях, ако разполагате с пълен бидон от L литра.

Точки: 10 точки, ако сложността $\mathrm{T}(n) = \mathrm{O}(n \,.\log n)$; 20 точки, ако $\mathrm{T}(n) = \mathrm{O}(n)$. Грешните алгоритми не носят точки. Бавните също. Демонстрирайте алгоритъма с пример. Анализирайте алгоритъма по време (ако липсва анализ, се отнемат 5 точки).

Решение: За да напълним възможно най-много съдове, трябва да изберем най-малките. Това може да стане чрез сортиране (напр. пирамидално сортиране) за време $\Theta(n \,.\, \log n)$ или чрез двоично търсене на разделителя с алгоритъма PICK за време $\Theta(n)$.

Зад. 3. Какво връща следният алгоритъм? **(20 точки)**
Дайте строго доказателство, например с инварианта.

```
   f(a, b: non-negative integers): integer
p ← 1
q ← b
while q > 0 do
   p ← p × a
   q ← q - 1
return p
```

Решение: Алгоритъмът връща a^{b}. Инварианта: $p = a^{b-q}$.

Зад. 1. Намерете времевата сложност на алгоритъма. **(10 точки)**

```
f(A[1...n]: array of integers): integer
if n = 1
    return A[1]
x ← f(A[1...n-1])
for i ← ⌊n/2⌋ + 1 to n
    if x < A[i]
        x ← A[i]
return x
```

Решение: Сложността удовлетворява рекурентното уравнение $\mathrm{T}(n) = \mathrm{T}(n-1) + \Theta(n)$. Първото събираемо е времето за рекурсивното извикване, а второто събираемо дава времето за всички други инструкции (нерекурсивната част). Чрез развиване или с помощта на характеристично уравнение намираме времевата сложност на алгоритъма: $\mathrm{T}(n) = \Theta\left(n^2\right)$.

Зад. 2. Известно е, че повече от половината елементи на числов масив $A[1 \ldots n]$ имат една и съща стойност. Намерете тази стойност.

Точки: 10 точки, ако сложността $\mathrm{T}(n) = \mathrm{O}(n \,.\, \log n)$; 20 точки, ако $\mathrm{T}(n) = \mathrm{O}(n)$. Грешните алгоритми не носят точки. Бавните също. Демонстрирайте алгоритъма с пример. Анализирайте алгоритъма по време (ако липсва анализ, се отнемат 5 точки).

Решение: Сортираме масива (например с пирамидално сортиране) за време $\Theta(n \,.\, \log n)$, след което връщаме средния елемент (медианата). Или намираме медианата с помощта на алгоритъма PICK за време $\Theta(n)$.

Зад. 3. Какво връща следният алгоритъм? **(20 точки)**
Дайте строго доказателство, например с инварианта.

```
f(a, b: positive integers): integer
p ← 0
q ← a
while q ≥ b do
    p ← p + 1
    q ← q - b
return p
```

Решение: Алгоритъмът връща $\left\lfloor \frac{a}{b} \right\rfloor$. Инварианта: $a = bp + q$.

Зад. 1. Намерете времевата сложност на алгоритъма. **(10 точки)**

```
f(A[1...n]: array of integers): integer
if n = 1
    return A[1]
x ← f(A[1...n-1]) + f(A[2...n])
for i ← ⌊n/2⌋ + 1 to n
    x ← x + A[i]
return x
```

Решение: Сложността удовлетворява рекурентното уравнение $\mathrm{T}(n) = 2\mathrm{T}(n-1) + \Theta(n)$. Първото събираемо е времето за рекурсивното извикване, а второто събираемо дава времето за всички други инструкции (нерекурсивната част). С помощта на характеристично уравнение намираме времевата сложност на алгоритъма: $\mathrm{T}(n) = \Theta\left(2^{n}\right)$.

Зад. 2. Масивът $A[1 \ldots n]$ съдържа цели положителни числа. Намерете три различни елемента (т.е. с различни индекси; обаче може да имат равни стойности), чийто сбор е 21.

Точки: 10 точки, ако сложността $\mathrm{T}(n) = \mathrm{O}(n^2 . \log n)$; 20 точки, ако $\mathrm{T}(n) = \mathrm{O}(n)$. Грешните алгоритми не носят точки. Бавните също. Демонстрирайте алгоритъма с пример. Анализирайте алгоритъма по време (ако липсва анализ, се отнемат 5 точки).

Решение: Сортираме масива (например с пирамидално сортиране) за време $\Theta(n . \log n)$, след което за всеки два елемента търсим трети, който ги допълва до сбор 21. Третия елемент намираме чрез двоично търсене, откъдето цялата времева сложност става $\Theta(n^2 . \log n)$: множителят n^2 е равен (по порядък) на броя на двойките от първите два елемента, а $\log n$ е времето за двоичното търсене.

По-бърз алгоритъм се постига с идеята на сортирането чрез броене. за време $\Theta(n)$ преброяваме по колко пъти се среща всяко цяло число от 1 до 19 (възможните събираеми на сбор 21). После за константно време проверяваме възможните варианти за образуване на желания сбор.

Зад. 3. Какво връща следният алгоритъм? **(20 точки)**
Дайте строго доказателство, например с инварианта.

```
f(a, b: non-negative integers): integer
p ← a
q ← 0
while p > 0 do
    p ← p - 1
    q ← q + b
return q
```

Решение: Алгоритъмът връща `ab`. Инварианта: `q = (a-p)b`.

Зад. 1. Намерете времевата сложност на алгоритъма. **(10 точки)**

```
f(A[1...n]: array of integers): integer
if n = 1
    return  A[1]
x ← f(A[1...⌊n/4⌋]) + f(A[⌊n/4⌋...⌊n/2⌋])
for i ← ⌊n/2⌋ + 1 to n
    for j ← ⌊n/2⌋ + 1 to n
        x ← x + A[i] × A[j]
return  x
```

Решение: Сложността удовлетворява рекурентното уравнение $\mathrm{T}(n) = 2\mathrm{T}\left(\frac{n}{4}\right) + \Theta\left(n^2\right)$. Първото събираемо е времето за двете рекурсивни извиквания, а второто събираемо дава времето за всички други инструкции (нерекурсивната част). От мастър-теоремата намираме времевата сложност на алгоритъма: $\mathrm{T}(n) = \Theta\left(n^2\right)$.

Зад. 2. В акционерно дружество участват n акционери. Масивът $A[1 \dots n]$ показва кой колко акции държи: $A[\,i\,] = k$ значи, че i-тият акционер държи k акции. Как да купим контролен пакет акции (т.е. повече от половината) с минимален брой сделки (т.е. да водим преговори с минимален брой акционери)?

Точки: 10 точки, ако сложността $\mathrm{T}(n) = \mathrm{O}(n \,.\, \log n)$; 20 точки, ако $\mathrm{T}(n) = \mathrm{O}(n)$. Грешните алгоритми не носят точки. Бавните също. Демонстрирайте алгоритъма с пример. Анализирайте алгоритъма по време (ако липсва анализ, се отнемат 5 точки).

Решение: Броят на сделките е минимален, ако ги сключим с най-богатите акционери. Сортираме ги по брой акции (напр. с пирамидално сортиране) за време $\Theta(n \,.\, \log n)$ или намираме разделителя за време $\Theta(n)$ чрез двоично търсене с алгоритъма PICK.

Зад. 3. Какво връща следният алгоритъм? **(20 точки)**
Дайте строго доказателство, например с инварианта.

```
f(a, b: positive integers): integer  // b ≥ 2
p ← 0
q ← a
while q ≥ b do
    p ← p + 1
    q ← q / b
return p
```

Отговор: Алгоритъмът връща $\lfloor \log_b a \rfloor$.

Инварианта: $q b^p = a$.