DAA Упражнение 4 (Коректност и сложност на алгоритми)

зад.1 / Даден е следният алгоритъм написан на псевдокод. Какво връща той? Докажете го формално и прецизно.

```
MYSTERY(A[1...n]: масив от цели числа)
1. for i ← 1 to n
2.     for j ← 1 to n-i
3.         if A[j] > A[j+1]
4.             swap(A[j], A[j+1])
```

Решение: Алгоритъмът MYSTERY сортира елементите на масива A в нарастващ ред.

Забележка: Важно е да се отбележи, че сортира елементите на масива A, а не че просто сортира. Проблемът е, че следният алгоритъм:

```
FAKESORT(A[1...n])
1. for i ← 1 to n
2.     A[i] ← i
```

също сортира масива, но оригиналните елементи се губят.

Инвариант(1)(Външен цикъл): При всяко достигане на ред 1 е изпълнено $A[n+2-i, \dots, n]$ се състои от $i-1$ най-големи елементи на входа в сортиран (нарастващ) вид.

Инвариант(2)(Вътрешен цикъл): При всяко достигане на ред 2 е изпълнено, че i-тият най-голям елемент на входа е максималният елемент в $A[j \dots n+1-i]$

Забележка: На контролно / изпит, ако няма да ви стигне времето, то доказването на външният цикъл (коректността му) ще ви донесе повече точки. Точно това ще направим в решението тук.

Доп, че инвариант(2) е изпълнен. Доказваме инвариант 1

База: При първото достигане на ред 1 имаме, че $i=1$. Пита се дали подмасивът $A[n+1 \dots n]$ се състои от 0 най-големи елемента на входа. Да, понеже $A[n+1 \dots n]$ е празен масив (т.е. изпълнено е в празния смисъл)

Поддръжка: Нека инвариант (1) е изпълнен за някое достигане на ред 1, което не е финално. От допускането, че инвариант (2) е изпълнен, то ~~при~~ в края на изпълнението имаме $j = n-i+1$ и инвариант (2) ни дава, че $A[n-i+1, \dots, n-i+1] = A[n-i+1]$ съдържа i-тият най-голям елемент на входа.

От инвариант(1) имаме, че $A[n+2-i,\dots,n]$ се състои от $i-1$ най-големи елементи на входа. Но $A[n+1-i]$ от инвариант(2) е "на мястото си" следователно $A[n+1-i,\dots,n]$ се състои от i най-големи елемента на входа. При следващо достигане на ред 1 имаме $i' = i+1 \leftrightarrow i = i'-1$ и след заместване получаваме $A[n+2-i',\dots,n]$ се състои от $i'-1$ най-големи елемента от входа. Следователно инвариантът се запазва спрямо новото i (i').

Терминация: При последното достигане на ред 1 $i = n+1$ и $A[1\dots n]$ се състои от n най-големи елемента в сортиран вид. Цикъла ще приключи, понеже на всяка стъпка i се инкрементира и е ~~фиксирано~~ ограничено отгоре от фиксирано n. ∎

Остана да докажем, че нашето допускане е правилно. Разглеждаме произволно изпълнение на вътрешния цикъл.

База: При първото достигане на ред 2, $j = 1$ и от доказаното за инвариант(1) имаме, че $A[n+2-i,\dots,n]$ се състои от $i-1$ най-големи елемента. Следователно i-тият по големина не е там и остава да е в $A[1\dots n+1-i]$ и $j=1$ значи е в $A[j\dots n+1-i]$.

Поддръжка: Нека инвариант(2) е изпълнен за някое достигане на ред 2, което не е финално. Следните случаи са изчерпателни:

I сл) $A[j] > A[j+1]$, тогава се разменят местата на $A[j]$ и $A[j+1]$ и е вярно, че max-а на $A[j\dots n+1-i]$ е max на $A[j+1\dots,n+1-i]$ (защото преди swap-а, ако $A[j]$ е max, то той ще остане част от подмасива след swap-а). Спрямо новото $j' = j+1 \leftrightarrow j = j'-1$ е изпълнено че i-тият най-голям елемент на входа е max елемент на $A[j',\dots,n+1-i]$

II сл) $A[j] \boxed{\leq} A[j+1]$, тогава нищо не се случва и очевидно $A[j]$ няма как да е max-а следователно той остава в $A[j+1\dots,n+1-i]$. Спрямо новото j'... (същото като I сл)

Терминация: При последно достигане на ред 2, $j = n+1-i$. Заместваме в инвариант(2) и получаваме, че i-тият най-голям елемент на входа е max. елемент в $A[n+1-i]$. Заедно с док. на инвариант(1), че $A[n+2-i\dots n]$ съдържа $i-1$ най-големи елемента на входа, то $A[n+1-i]$ е "на мястото си". Очевидно цикъла ще терминира. ∎

Изследването на сложността на алгоритми ще правим, като търсим каква е сложността по време (в някои случаи и по памет) в най-лошия случай, но ще искаме тя да е възможно най-точна/ниска. Например ако даден алгоритъм е $O(n)$, то той също е $O(n^2)$, но ще записваме, че е $O(n)$. За удобство ще ползвам конвенцията, че $T(n)$ е сложност по време и $M(n)$ е сложност по памет (тоест колко допълнителна памет използваме), но тези конвенции не са одобрени, така че на контролно/изпит може да не се зачетат.

Всяко число заема $\Theta(1)$ памет, всички операции с тях са $\Theta(1)$ $(+,/,*,\ldots)$, if е $\Theta(1)$, една итерация на цикъл е $\Theta(1)$ и т.н.

Някои полезни суми: $\sum_{i=1}^{n} 1 = n = \Theta(n)$; $\sum_{i=1}^{n} \underbrace{\Theta(1)}_{const} = \Theta(n)$; $\sum_{i=c}^{n} 1 = n-c+1 = \Theta(n)$

$\sum_{i=c}^{n} \Theta(1) = \Theta(n)$; $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$; $\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$

$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$; $\sum_{i=1}^{n} \Theta(i^2) = \Theta(n^3)$; $\sum_{\substack{i=1 \\ i+=k}}^{n} 1 \approx \frac{n}{k} = \Theta(n)$

$\sum_{\substack{i=1 \\ i+=k}}^{n} \Theta(1) = \Theta(n)$; $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\lg n)$; $\sum_{i=1}^{n} i^{\alpha} = \begin{cases} \Theta(n^{\alpha+1}), & \alpha > -1 \\ \Theta(\lg n), & \alpha = -1 \\ \Theta(1), & \alpha < -1 \end{cases}$

$\sum_{i=a}^{b} f(i) \Longleftrightarrow \sum_{i=1}^{b} f(i) - \sum_{i=1}^{a-1} f(i)$

зад. 2/ Условието на останалите задачи от това упражнение е следното: Даден е следният алгоритъм. Изследвайте сложността му по време в най-лошият случай.

```
ADD1 (n ∈ N+)
1. a ← 0
2. for i ← 1 to n
3.     for j ← 1 to n
4.         a ← a + 1
5. return a
```

Решение: $\underbrace{\Theta(1)}_{1,5 \text{ ред}} + \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} \underbrace{\Theta(1)}_{4 \text{ ред}}}_{2,3,4 \text{ ред}} = \Theta(1) + \sum_{i=1}^{n} n = \Theta(1) + n\sum_{i=1}^{n} 1 =$

$= \Theta(1) + n^2 = \Theta(n^2)$

зад.3/ EXAMPLE2($n \in \mathbb{N}^+$)

```
1. for i ← 1 to n
2.     for j ← 1 to i
3.         print("Hello")
4. print("Bye")
```

Решение: $\underbrace{\Theta(1)}_{4\text{ ред}} + \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{i} \underbrace{\Theta(1)}_{3\text{ ред}}}_{1,2,3\text{ ред}} = \Theta(1) + \sum_{i=1}^{n} i =$

$= \Theta(1) + \Theta(n^2) = \Theta(n^2)$

зад.4/ EXAMPLE3($n \in \mathbb{N}^+$)

```
1. a ← 0
2. for i ← 1 to n
3.     for j ← i+1 to n
4.         for k ← 1 to j
5.             a ← a+1
6. return a
```

Решение: $\underbrace{\Theta(1)}_{1,6\text{ ред}} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{j} \Theta(1) =$

$= \Theta(1) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} j = \Theta(1) + \sum_{i=1}^{n} \left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j \right) =$

$= \Theta(1) + \sum_{i=1}^{n} \frac{n(n+1)}{2} - \sum_{i=1}^{n} \frac{i(i+1)}{2} =$

$= \Theta(1) + \frac{n.n(n+1)}{2} - \frac{1}{2}\sum_{i=1}^{n} i^2 - \frac{1}{2}\sum_{i=1}^{n} i =$

$= \Theta(1) + \frac{n^3}{2} + \frac{n^2}{2} - \frac{n(n+1)(2n+1)}{12} - \frac{n(n+1)}{4} = \Theta(n^3)$ (Разкрийте сами скобите)

зад.5/ GRAHAMSCAN(A[1...n]: масив от цели числа)

```
1. S ← празен стек с pop/push/top за Θ(1)
2. S.push(A[1])
3. S.push(A[2])
4. for i ← 3 to n
5.     while S.NotEmpty and P(S.top())
6.         S.pop()
7.     S.push(A[i])
```

P е някакъв предикат, например >10, който се смята константно.

Решение: Ако забележим, че ред 4 ще се изпълни n на брой пъти и ред 5 ще се изпълни в най-лошият случай n пъти, то тогава излиза, че сложността е $\Theta(n^2)$. Обаче, ако съобразим, че един елемент от масива може само веднъж да е добавен/премахнат, то сложността на алгоритъма е $\Theta(n)$.

зад.6/ EXAMPLE4($n \in \mathbb{N}^+$)

```
1. a ← 0
2. for i ← 1 to n
3.     for j ← 1 to n
4.         if i = j
5.             for k ← 1 to n
6.                 a ← a+1
7. return a
```

Решение: $n.\sum_{k=1}^{n} \Theta(1) + (n^2-n).\Theta(1) + \Theta(1) =$

$= n^2 + \Theta(n^2) = \Theta(n^2)$

Понеже n пъти имаме $i=j$ и останалите n^2-n пъти имаме само проверката на ред 4 да е лъжа.

зад. 7/ EXAMPLE5 ($n \in \mathbb{N}^+$)

```
1. for i ← 1 to n
2.     for j ← 1 to n with step i
3.         print("Hello")
```

Решение: $\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j+=i}}^{n} \approx \sum_{i=1}^{n} \frac{n}{i} = n \cdot \sum_{i=1}^{n} \frac{1}{i} = n \cdot \theta(\lg n) = \theta(n \lg n)$

зад. 8/ EXAMPLE6 ($n \in \mathbb{N}^+$)

```
1. for i ← 1 to n
2.     for j ← 1 to n with step i
3.         for k ← 1 to n with step i
4.             print("Hello")
```

Решение: $\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j+=i}}^{n} \sum_{\substack{k=1 \\ k+=i}}^{n} 1 \approx \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j+=i}}^{n} \frac{n}{i} = \sum_{i=1}^{n} \left(\frac{n}{i} \sum_{\substack{j=1 \\ j+=i}}^{n} 1 \right) = \sum_{i=1}^{n} \frac{n^2}{i^2} =$

$= n^2 \sum_{i=1}^{n} \frac{1}{i^2} = \theta(n^2)$