Зад.1 Selection sort

Докажете коректност и изведете скоростта на алгоритъма.

```
sort(A[1...n]: array of numbers): void
for i ← 1 to n-1 do          ← инв.①
    min ← i
    for j ← i+1 to n do      ← инв.②
        if A[j] < A[min] then
            min ← j
    swap(A[i], A[min])
```

Инв.①: При всяко достигане на проверката $i \leq n-1$ (?) имаме, че масива $A[1,\dots,(i-1)]$ е сортиран и в тях се съдържат $(i-1)$-те най-малки елементи на масива. $i \in \{1,\dots,n\}$

Инв.②: При всяко достигане на проверката $j \leq n$ (!) имаме, че $A[min] = \min\{A[i],\dots,A[j-1]\}$
$j \in \{i+1,\dots,n+1\}$

Вложени цикли => вложени д-ва.
$O(n^2)$

Инварианта ①:
При всяко достигане до проверката $i \le n-1$ имаме, че масива $A[1 \ldots (i-1)]$ е сортиран и в него се намират $(i-1)$-те най-малки елемента на масива $A[1 \ldots n]$ за $i \in \{1, \ldots, n\}$

Инварианта ②:
При всяко достигане на проверката $j \le n$ имаме, че
$$A[min] = \min A[i \ldots (j-1)] \text{ за } j \in \{i+1, \ldots, n+1\}$$

Сега ще направим доказателствата в термините на терминологията на курса т.е. индукциите ще бъдат разделени на три части:
① База; ② Поддръжка; ③ Терминация.
"база" "ИП + стъпка ...." "посл. път."
Към терминацията ще добавим и защо завършва въобще алгоритъма. Което видяхме при

BinarySearch имаше нужда от допълнителни клаузи инв., за да сметнем и асимптотичната сложност, но специално тук е сравнително просто.
Сега да докажем инв. ① и ②
Чрез вложени индукции ще докажем коректността на инвариантите

База: Имаме $i=1$. Тогава
$A[1\ldots(i-1)] = A[1\ldots0] = A[\,]$ празен масив и знаем, че на празния масив елементите му изпълняват което свойство искаме да имат.

Поддръжка ①: Нека предположим, че е в сила инвариантата ① за някое нейно изпълнение, което не е последното за външния for и сега ще докажем че в него е в сила инвариантата ②:

База: Имаме $min = i$, $j = i+1$.
Тогава $A[i\ldots(j-1)] = A[i\ldots i] = A[i]$
и $A[min] = A[i] = \min A[i]$

Поддръжка: Нека за някое изпълнение на вътрешния for, което не е последно е в сила инвариантата т.е $j \in \{i+1, \ldots, n\}$:

$$A[min] = \min A[i \ldots (j-1)]$$

Искаме да докажем, че при следващото достигане на проверката (?) $j \le n$ (?), т.е. вече за $j_{new} = j+1$ ще имаме спрямо изменената стойност на j:

$$A[min_{new}] = \min A[i \ldots (j_{new}-1)]$$

Следим изпълнението на алгоритъма ред по ред какво се случва с променливите, от чиито ст-ти зависи верността на инвариантата:

Сл. 1 $A[j] < A[min]$

Тогава $min_{new} = j$ и при достигане след това до проверката (?) $j \le n$ (?) за $j_{new} = j+1$ имаме:

$A[min_{new}] = A[j]$

От И.П. имаме, че за $min \in \{i, \dots, j-1\}$,

то $A[min] = \min A[i\dots(j-1)]$ и

сега имаме, че $A[j] < A[min] = \min A[i\dots(j-1)]$

т.е. $A[j] = A[min_{new}] = \min A[i\dots j] =$

$= \min A[i\dots(j_{new}-1)]$.

Сл.2 $A[j] \geq A[min]$

Тогава от ИП имаме, че

$A[min_{new}] = A[min] = \min A[i\dots(j-1)]$ и

сега $A[min] \leq A[j]$ и значи

$A[min_{new}] = A[min] = \min A[i\dots j] =$

$= \min A[i\dots(j_{new}-1)]$.

<u>Терминация</u>: Нека $j = n+1$. Тогава от инварианатата имаме, че

$$(\star)\left\{\begin{aligned} A[min] &= \min A[i\dots(j-1)] = \\ &= \min A[i\dots(n+1-1)] = \\ &= \min A[i\dots n] \end{aligned}\right.$$

Това твърдение го означаваме като $(\star)$, защото ще ни трябва следнапка.

Продължаваме с поддръжката на инварианта (1).

Имаме (ИП), което гласи за i:

$A[1 \dots (i-1)]$ е сортиран и

тук са $(i-1)$-те най-малки елементи на $A[1 \dots n]$.

От (*) имаме, че:

$A[min] = \min A[i \dots n]$ и при

изпълнението на следващия ред разменяме текущото ст-т на $A[i]$ с този на $A[min]$ т.е.

$A[i] = \min A[i \dots n]$.

Тогава комбинираме с ИП и получаваме, че $A[1 \dots i]$ е сортиран и съдържа $(i+1)$-най-малки ел. на $A[1 \dots n]$

(използваме $A[1 \dots (i-1)]$ е сортиран и те са $(i-1)$-те най-малки елементи на масива $A[1 \dots n]$, а $A[i]$ е най-малкия елемент на масива $A[i \dots n]$ и със сигурност $A[i] \geq A[k]$ за $k \in \{1, \dots, (i-1)\}$, така че е в сила)

Спрямо $i_{new} = i+1$, то имаме:
$A[1..(i_{new}-1)]$ е сортиран и съдържа $(i_{new}-1)$-те най-малки елемента на масива $A[1...n]$.

Терминация: За $i = n$ имаме от инвариантата, че
$A[1...(i-1)] = A[1...(n-1)]$
е сортиран и съдържа $(n-1)$-те най-малки елемента на $A[1...n]$.
Т.е. $A[n]$ е най-големият елемент на масива $A[1...n]$ и така $A[1...n]$ го получаваме сортиран след края на изпълнението на външния цикъл.

Сега пълната коректност т.е., че завършва е окей, защото имаме вложени цикли по естествени числа, които не се променят по време на изпълнение.
Сега сложност (асимптотична)

е $\Theta(n^2)$, но нека да го разпишем по-формално:
Имаме $(n-1)$ изпълнения на външния цикъл и спрямо текущото изпълнение на външния цикъл имаме $(n-i)$ изпълнения на вътрешния цикъл т.е. имаме:

$$\sum_{\substack{i=1\\ i=i+1}}^{n-1} \sum_{\substack{j=i+1\\ j=j+1}}^{n} 1 = \sum_{\substack{i=1\\ i=i+1}}^{n-1} \Big(\underbrace{\sum_{j=1}^{n} 1}_{n} - \underbrace{\sum_{j=1}^{i} 1}_{i} \Big) =$$

$$= \sum_{i=1}^{n-1} (n-i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} 1 =$$

$$= (n-1).n - \frac{(n-1).n}{2} = \frac{1.n^2}{2} - \frac{1.n}{2} \simeq n^2$$

т.е. имаме, че сложността на алгоритъма по време е $\Theta(n^2)$.

Зад / Kadane: максимална сума на подмасив

kadane(A[1..n] : array of numbers) : number

1) $localMax \leftarrow 0$

2) $globalMax \leftarrow -\infty$

3) for $i \leftarrow 1$ to n do ← Инв.

4) $localMax \leftarrow max(A[i] + localMax, A[i])$

5) $globalMax \leftarrow max(localMax, globalMax)$

6) return globalMax

За $max(a, b) = a$, ако $a > b$ и $max(a, b) = b$ иначе

Инв: При всяко достигане на проверката ? $i \leq n$? имаме, че в localMax се съдържа най-голямата сума на подмасив, завършващ с $A[i-1]$, а в globalMax се съдържа най-голямата сума сред тези завършващи в $A[0], \dots, A[i-1]$. За $i \in \{1, \dots, n+1\}$.

Сложност: $\Theta(n)$.

Започваме с доказателството на инварианта:

База: Имаме, че $i = 1$, $localMax = 0$, $globalMax = -\infty$ и това инвариантата тривиално е изпълнена за несъществуващи подмасиви $A[1..0]$.

<u>Поддръжка</u>: Допускаме, че е в сила за някое влизане в цикъла, което не е последно.

Сега искаме за следващото достигане на проверката ($i \le n$?) да е в сила инвариантата за променените ст-ти на променливите, от които зависи верността ѝ.

Гледаме ред по ред какво се случва.

$localMax_{new} = max(A[i], localMax + A[i])$

(Сл. 1) $localMax_{new} = A[i]$

Тогава

$$globalMax_{new} = max(localMax_{new}, globalMax)$$

(Сл. 1.1) $globalMax_{new} = localMax_{new}$

Тогава имаме, че

$$localMax + A[i] \le A[i] \text{ и}$$
$$globalMax < A[i]$$

От ИП имаме, че в localMax
се съд. най-голямата сума на
подмасива $A[1\ldots(i-1)]$, а в
globalMax най-голямата сума на
един от подмасивите $A[1]$, $A[1\ldots2]$, ...
..., $A[1\ldots(i-1)]$.

От $localMax + A[i] \leq A[i]$
значи сумата с единствен
елемент $A[i]$ е максимална
за подмасива $A[1\ldots i]$, а
от $globalMax < A[i]$, имаме че
най-голямата сума на подмасива
$A[1\ldots i]$ т.е. само $A[i]$ е по-голямо
от най-големите суми на подмасиви
завършващи съответно с
$A[1]$, $A[2]$, ..., $A[(i-1)]$.
Правилно сме актуализирали
стойностите и спрямо новото
$i_{new} = i+1 \Leftrightarrow i = i_{new} - 1$
инвариантата е вярна.

(сл. 1.2) $globalMax_{new} = globalMax$

Със същите разсъждения имаме, че $localMax_{new} = A[i]$ е окей - най-голямата сума на подмасив завършващ с $A[i]$. Имаме, че $localMax_{new} \leq globalMax$ т.е. не сме подобрили глобално състоянието на търсеното или за най-голяма сума на подмасив. Отново инвариантата ще бъде в сила.

сл. 2) $localMax_{new} = localMax + A[i]$

сл. 2.1) $globalMax_{new} = localMax_{new}$.

т.е. $A[i] < localMax + A[i]$ и

$globalMax < localMax + A[i]$

Освен, че сме подобрили локалното състояние на търсеното, ние сме намерили и по-добро глобално. Продължаваме предна сума на някой от подмасивите завършващ с $A[1], A[2], \ldots$ или $A[(i-1)]$. Съответно спрямо

изменените ст-ти инвариантата
отново е в сила.

сл. 2.2 globalMaxNew = globalMax
т.е. globalMax ≥ localMax + A[i]
Намерили сме по-добро
локално състояние т.е.
намерили сме най-голямата
сума на подмасива завършващ
с A[i], която в този случай
превишава предната сума (на
като в случай 1), но не сме
подобрили глобалната. Спрямо
изменените ст-ти инвариантата
отново е в сила.

Терминация: За $i = n+1$ имаме от
инвариантата, че в globalMax
имаме най-голямата сума на
подмасив завършващ на
A[1], A[2], ..., A[n] и
след това я връщаме обратно
този ст-т в края на изпълнението

на алгоритъма.

Съответно имаме частична коректност. Пълна коректност, че завършва е в сила, т.к. имаме цикъл по естествено число.

Асимптотична сложност е $\Theta(n)$.

```
do(A[1...n] : array of numbers): set
i ← 1
while A[i] ≤ 0 and i ≤ n do          ← ИНВ ①
    i ← i+1
if i = n+1
    return EmptySet
j ← i
temp ← A[i]                                   ИНВ ②
max ← 0, maxi ← 0, maxj ← 0                ←
while j ≤ n
    if temp > max
        max ← temp
        maxi ← i
        maxj ← j
    j ← j+1
    temp ← temp + A[j]
    if temp ≤ 0
        temp ← 0
        i ← j+1
return (maxi, maxj)
```

Алгоритъмът връща началната и крайната точка на поредицата от елементи с най-голяма положителна сума.

Сега най-лошият случай на този алгоритъм ще е $O(n)$, защото while цикълът инкрементира j с 1-ца докато не стигнем n, а когато i=1 получаваме n итерации.

Първият while цикъл прави проверка дали има поне едно положително число и откъде от него да започнем алгоритъма. Ако всички числа в масива са ≤ 0, то няма какво да вършим нататък и връщаме $\emptyset$. Ако във входящият масив има поне едно положително число, вече става интересно и индекса, на който се намира го записваме в i.

Да формулираме тогава инварианти:

Инвариант ①:

При всяко достигане до проверката на предиката в while цикъла на ред ②, то в подмасива $A[1 \ldots (i-1)]$ има само неположителни числа.

Инвариант ②:

При всяко достигане до проверката на предиката в while цикъла на ред ②, то сумата на подредицата $A[maxi \ldots maxj]$ в max е най-голямата положителна в масива $A[1 \ldots (j-1)]$, докато сумата на подредицата $A[i \ldots (j-1)]$ в temp е най-голямата неотрицателна сума в масива $A[1 \ldots (j-1)]$.

Проверката на инвариант ① е проста, така че нека се съсредоточим в/у инвариант ②:

База: max = 0, temp = A[i], maxi = 0, maxj = 0,
j = i

От инв.1) имаме, че или i е индекса на първия положителен елемент в масива A[1...n] (случаят, когато е n+1 се хваща от предния if и тогава завършваме алгоритъма с return ∅).

каквото и да кажем за елем. на празен масив е в сила

Тогава max е най-голямата полож. сума на подредицата A[0...0] = A[ ] ✓ и temp = A[i] > 0 е най-голямата неотр. сума на подредиц. A[0...0] = A[ ] ✓

Поддръжка: Нека е в сила инвариантата за някое влизане, което не е последно. При следващото достигане на проверката j ≤ n? искаме тя отново да е в сила за изменените ст-ти на променливите. Гледаме ред по ред какво се случва.

Сл. 1 Ако $temp > max$

Тогава $max_{new} = temp$, $maxi_{new} = i$, $maxj_{new} = j$

$j_{new} = j+1$ и $temp' = temp + A[j]$

Сл. 1.1 $temp' \leq 0$

$temp_{new} = 0$

$i_{new} = j_{new} + 1$

Сега $max_{new} = temp > max$ но допускане, а от ИИ $max > 0$ значи $max_{new} = temp > max > 0$ и $maxi_{new} = i$, $maxj_{new} = j$.

Откъдето от ИИ в max е най-голямата неотр. сума в $A[1 \ldots (j-1)]$ на подрез. $A[maxi \ldots maxj]$, а сега спрямо новите индекси $A[maxi_{new} \ldots maxj_{new}]$ е в сила едно част от инвариантата

И т.к. $temp > max > 0$, то значи сме актуализирали границите на най-голямата полож. сума на

подредица с граници 1 и j
$A[1\ldots(j_{new}-1)] = A[1\ldots j]$.
Всеки новото $temp_{new} = 0$ и
спряно $i_{new} = j_{new}+1$
имаме, че $temp_{new}$ след. най-голямата
неотриц. сума на подредицата:
$A[i_{new}\ldots j_{new}] = A[(j_{new}+1)\ldots j_{new}] = A[\,] = 0$
Сл. 1.2 $temp' > 0$
За max_{new} е свиетото, а
за $temp' = temp + A[j]$, че
е най-голямата неотр. сума с
подредица $A[i_{new}\ldots(j_{new}-1)]$ за
$i_{new} = i$ и $j_{new} = j+1$ спряно
$A[1\ldots j] = A[1\ldots(j_{new}-1)]$.
Сл. 2 Ако $temp \leq max$, тоже
променяме $max, maxi, maxj$
$j_{new} = j+1$ и $temp' = temp + A[j]$
Сл. 2.1 $temp' \leq 0$
$temp_{new} = 0$
$i_{new} = j_{new}+1$
От ИП за $max_{new} = max$, $maxi_{new} = maxi$,

$maxj_{new} = maxj$, а за $temp_{new}$ са аналогични разсъжденията.

Сл. 2.2 $temp' > 0$

Както предният случай за max, а за $temp'$ както в 1.2.

<u>Терминация</u>: За $j = n+1$ от инв. имаме, че в max е сумата на подредицата $A[maxi \dots maxj]$ на най-голямата положителна сума на подредици в масива $A[1 \dots n]$ и точно $\langle maxi, maxj \rangle$ наредената двойка в края.

Точно това искахме и за връщане.

Ясно е, че завършва алгоритъма, а сложността му коментирахме по-рано.

Намиране асимптотична сложност на рекурентни уравнения

I-ва част: чрез решаване на характеристични уравнения (които са с рекурсия и постоянни коеф.) развиване, заместване и др.

Зад. 1

$T(n) = 8.T(n-1) - 15T(n-2) + (8n+5)4^n - 3^n + 2n^3$

$T(n) = 8.T(n-1) - 15.T(n-2) + (8n^1 + 5n^0).4^n - n^0.3^n + 2.n^3.1^n$

Решаване хомогенната част на уравнението:

$x^n = 8.x^{n-1} - 15x^{n-2} \quad // : x^{n-2} \neq 0$

$x^2 = 8.x - 15$

$D = 8^2 - 4.15 = 64 - 60 = 4$

$x_{1,2} = \frac{8 \pm 2}{2} \begin{matrix} \nearrow 5 \\ \searrow 3 \end{matrix} \Big\} \to \{3, 5\}_m$ multiset

Сега нехомогенната част да извлечем корени (броят им зависи от степените на n т.е. ще бъде $deg(n)+1$)

$(8.n^1 + 5.n^0).4^n - n^0.3^n + 2n^3.1^n$

$\{4, 4, 3, 1, 1, 1, 1\}_m$

т.е. корените на уравнението ни са:

$\{3,5\}_m \cup \{4,4,3,1,1,1,1\}_m =$

$\{3,5,4,4,3,1,1,1,1\}_m$

Общият вид на уравнението е:

$T(n) = (c_1 + c_2.n + c_3.n^2 + c_4 n^3).1^n +$
$(c_5 + c_6.n).3^n + (c_7 + c_8.n)4^n +$
$c_9.5^n$ за $\underline{c_9 > 0}$

и $T(n) \asymp 5^n$

зад. 2 $T(n) = T(n-1) + \sqrt[5]{n}$

т.е. $T(n) = T(n-1) + \underline{n^{1/3}.1^n}$

Не става с характеристично урe заради дробнята час. Ще пробваме да развием уравнението:

$T(n) = T(n-1) + n^{1/3} = T(n-2) + (n-1)^{1/3} + n^{1/3} =$
$= T(n-3) + (n-2)^{1/3} + (n-1)^{1/3} + n^{1/3} = \dots =$
$= T(0) + \underbrace{1^{1/3}}_{(n-(n-1))^{1/3}} + \underbrace{2^{1/3}}_{(n-(n-2))^{1/3}} + \dots + (n-1)^{1/3} + n^{1/3} =$
$= \underbrace{T(0)}_{const} + \sum_{i=1}^{n} i^{1/3} \asymp \sum_{i=1}^{n} i^{1/3} \asymp n^{1/3+1} = n^{4/3}$

$1/3 > -1$

т.е. $T(n) \asymp n^{4/3}$

Приложение:

(*) $\sum_{i=1}^{n} i^{k} \asymp \begin{cases} n^{k+1}, & k > -1 \\ \log n, & k = -1 \\ 1, & k < -1 \end{cases}$

(*) $\sum_{\substack{i=1 \\ i=i+k}}^{n} 1 = \left\lceil \frac{n}{k} \right\rceil \approx \frac{n}{k}$ (*) $\sum_{i=a}^{b} 1 = b-a+1$

Зад 3 $T(n) = 2.T(n-1) - T(n-2)$

От х-е $x^n = 2x^{n-1} - x^{n-2} \;//: x^{n-2} \neq 0$

$x^2 = 2x - 1$, защото $D = 4 - 4 = 0$

$\{1, 1\}_m$

Общо решение:

$T(n) = (c_1 + c_2.n).1^n \asymp n$

за $c_2 > 0$

Зад 4 $T(n) = T(n-1) + \frac{4}{n^{10}}$

Не става с харак. ур-е заради отриц. степени.

Развиване ур-ето:

$T(n) = T(n-2) + \frac{4}{(n-1)^{10}} + \frac{4}{n^{10}} = \dots =$

$$= \underbrace{T(0)}^{const} + 4 \cdot (1^{-10} + 2^{-10} + \dots + n^{-10}) \asymp$$

$$\sum_{i=1}^{n} i^{-10} \asymp \underline{1}$$

т.е. $\underline{T(n)} \asymp 1$

ограничена ф-я.

Зад (5) $T(n) = T(n-1) + 1$

Сразписване:

$$T(n) = T(n-2) + 1 + 1 = \dots = T(0) + \underbrace{1 + \dots + 1}_{n \text{ пъти}} =$$

$$= \underbrace{T(0)}_{const} + \sum_{i=1}^{n} 1 \asymp n - 1 + 1 = \underline{n}$$

Зад (6) $T(n) = T(n-1) + 1/n$

Отрицателна степен $\frac{1}{n}$ -> разписване

$$T(n) = T(n-2) + \frac{1}{n-1} + \frac{1}{n} = \dots =$$

$$= T(0) + \frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n} = T(0) + \sum_{i=1}^{n} \frac{1}{i} \asymp$$

$$\sum_{i=1}^{n} \frac{1}{i} \asymp \log n \quad \text{т.е.} \quad T(n) \asymp \log n$$

Зад (7) $T(n) = \underline{n} \cdot T(n-1)$

Не става с харак. ур-е заради променлив коефициент n -> разписване:

$$T(n) = n \cdot (n-1) \cdot T(n-2) = n \cdot (n-1) \cdot (n-2) \cdot T(n-3) = \dots$$

$$= n\cdot(n-1)\cdot(n-2)\dots 1\cdot \overbrace{T(0)}^{const} \asymp n!$$

т.е. $T(n) \asymp n!$

Зад. 8 $T(n) = 2\cdot T(n-1) + \frac{1}{n}$ $= n^{-1}$

Развиване:

$$T(n) = 2^2\cdot T(n-2) + \frac{2}{n-1} + \frac{1}{n} = 2^3\cdot T(n-3) + \frac{4}{n-2} + \frac{2}{n-1} + \frac{1}{n} = 2^n\cdot T(0) + \frac{1}{n} + \frac{2}{n-1} + \frac{4}{n-2} + \dots + \frac{2^{n-1}}{1} =$$

$$= 2^n\cdot T(0) + 2^n\cdot \sum_{i=1}^{n} \frac{1}{i\cdot 2^i}$$

$$\sum_{i=1}^{n} \frac{1}{i\cdot 2^i} \le \sum_{i=1}^{n} \frac{1}{2^i} \le \sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

$$2^n\cdot \sum_{i=1}^{n} \frac{1}{i\cdot 2^i} \le 2^n\cdot 1 = 2^n$$ т.е. $T(n) \asymp 2^n$

Зад. 9 $T(n) = \frac{n}{n+1}\cdot T(n-1) + 1$

С развиване $T(n)$ получаваме:

$$T(n) = \frac{\cancel{n}}{n+1}\cdot\frac{n-1}{\cancel{n}}\cdot T(n-2) + \frac{n}{n+1} + 1 =$$

$$= \frac{\cancel{n-1}}{n+1}\cdot\frac{n-2}{\cancel{n-1}}\cdot T(n-3) + \frac{n-1}{n+1} + \frac{n}{n+1} + 1 = \dots$$

$$= \frac{1}{n+1} \cdot T(0) + \frac{n+1}{n+1} + \frac{n}{n+1} + \frac{n-1}{n+1} + \dots + \frac{2}{n+1} =$$

$$= \frac{1}{n+1} \cdot T(0) + \frac{1}{n+1} \cdot \sum_{i=2}^{n+1} i = \qquad = \sum_{i=1}^{n+1} i - 1$$

$$= \frac{1}{n+1} \cdot \underbrace{T(0)}_{\text{const}} + \frac{1}{n+1} \cdot \left(\frac{(n+1) \cdot (n+2)}{2} - 1 \right) \simeq \underline{n}$$

T.e $T(n) \simeq n$