`Rearrange`($A[1..n]$: `array of integers`; ℓ, r: `indices in` A; p: `pivot index in` $[\ell..r]$)
@1 $x \leftarrow A[p]$
@2 $swap(A, p, r)$
@3 $i \leftarrow \ell$
@4 $j \leftarrow r - 1$
@5 `while` $i < j$ `do`
@6   `if` $A[i] \geq x$ `and` $A[j] < x$ `then`
@7 $swap(A, i, j)$
@8   `if` $A[i] < x$ `then`
@9 $i \leftarrow i + 1$
@10   `if` $A[j] \geq x$ `then`
@11 $j \leftarrow j - 1$
@12 `done`
@13 `if` $A[i] < x$ `then` $i \leftarrow i + 1$
@14 $swap(A, i, r)$
@15 `return` i

Лема 0.1. *При всеки вход от масив $A[1..n]$, индекси $1 \leq \ell \leq r \leq n$ и индекс p:*

1. *$Rearrange(A, \ell, r, p)$ завършва за време $\Theta(r - \ell + 1)$.*
2. *ако A' е масивът A след края на процедурата, а i е нейният резултат, то:*

$$A'[1..\ell - 1] \circ A'[r + 1..n] = A[1..\ell - 1] \circ A[r + 1..n] \text{ и } A'[\ell..r] \text{ е пермутация на } A[\ell..r]$$

като за всяко $s \in [\ell..r]$ е в сила, че $A'[s] < A[p] = A'[i]$ точно когато $s < i$.

Доказателство. Ясно е, че след @2 A е модифициран до $\hat{A}[r] = x$, $\hat{A}[p] = A[r]$.

Да забележим, че при всяко изпълнение на тялото на while-цикъла се изпълнява поне един от двата реда @8 или @10. Наистина, ако на ред @5 не е изпълнено условието $A[i] \geq x$ AND $A[j] < x$, то е вярно неговото отрицание, тоест поне едно от двете условия $A[i] < x$ и $A[j] \leq x$ е изпълнено. Но това са точно условията, при които се изпълняват ред @8 и ред @10 съответно. Ако условието на ред @5 е изпълнено, то след изпълнението на ред @6 за изменения масив $\widetilde{A}$ е в сила, че $\widetilde{A}[i] = A[j] < x$ и $\widetilde{A}[j] = A[i] \geq x$. Следователно се изпълняват и двата реда @8 и @10.

С това показахме, че при всяко изпълнение на тялото на цикъла i се увеличава на ред @8 или j намалява на ред @10. Тоест разликата $j - i$ намалява поне с 1 (и най-много с 2) при всяка итерация на цикъла. Тъй като след ред @3, $j - i = r - 1 - \ell$, то броят на итерациите на цикъла е $\Theta(r - \ell)$, а оттук и сложността на функцията е $\Theta(r - \ell + 1)$.

От горното разсъждение, тъй като $r > j > i \geq \ell$, когато се изпълнява тялото на цикъла, то промените на масива A засягат само елементите му на позиции $i, j \in [\ell, r - 1]$, като единствените промени са транспозиции на такива елементи. Следователно:

$$A'[1..\ell - 1] \circ A'[r + 1..n] = A[1..\ell - 1] \circ A[r + 1..n] \text{ и } A'[\ell..r] \text{ е пермутация на } A[\ell..r].$$

Накрая, ако с i_t, j_t и A_t означим стойностите на i, j и състоянието на масива A преди t-тата итерация на цикъла, дефинираме свойството $I(t)$ като:

$$I(t) : \max A[\ell..i_t - 1] < x = A[r] \leq \min A[j_t + 1..r - 1].$$

С индукция по t непосредствено се проверява, че $I(t)$ е вярно за всяко t. В частност, когато $i_t \geq j_t$ е изпълнено $I(t)$. Тогава, ако $A_t[i_t] < x$, то $i_t \leq j_t$, защото иначе, $j_t + 1 \leq i_t$ и от $I(t)$ щяхме да имаме, че $A_t[i_t] \geq x$, което не е вярно. Следователно, ако $A[i_t] < x$, то $i_t = j_t$ и тогава $A[i_t + 1] \geq x$. Така след ред @13, $\max A[\ell..i - 1] < x \leq \min A[\ell..r - 1]$ и $A[r] = x$.

Оттук, като отчетем ред @14 получаваме и резултатът за A'. ☐

`QuickSelectK`($A[1..n]$: `array of integers`; s, t:`indices in` A; k: `integer in the range` $[1..t - s + 1]$)
@1 $p' \leftarrow \mathcal{U}(\{1, \ldots, t - s + 1\})$
@2 $p \leftarrow p' + s - 1$
@3 $\ell \leftarrow Rearrange(A, s, t, p)$
@4 `if` $s + k - 1 = \ell$ `return A[p]`
@5 `if` $\ell >\ s + k - 1$ `then`
@6   `return` $QuickSelectK(A, s, \ell - 1, k)$
@7 `else`
@8   `return` $QuickSelectK(A, \ell + 1, t, k + s - \ell)$

Лема 0.2. *При вески масив от числа $A[1..n]$ и индекси $1 \leq s \leq t \leq n$ и $1 \leq k \leq t-s+1$, $QuickSelectK(A,s,t,k)$ има следните свойства:*

1. *Коректно намира k-тия по големина елемент на $A[s..t]$.*
2. *При предположение, всички числа в $A[1..n]$ са различни и че различните извиквания на $\mathcal{U}$ генерират в съвкупност независима случайни величини, то очакваното време за $QuickSelectK(A,s,t,k)$ е $O(t-s)$.*

Доказателство. Доказателството на коректността следва с индукция по $t-s$. При $t=s$, $j=0$. Тогава трябва да е вярно, че $k=1$, а от предишната лема имаме, че $\ell = s$. Сега е ясно, че резултатът е коректен на ред @4.

При $t > s$, имаме следното. Първо, нека $q = A[p]$ след ред @2. Тогава, $q \in A[s..t]$ и след @3 е в сила, че $A[\ell] = q < A[\ell+1]$, като всички елементи в $A[s..\ell]$ не надминават q, а всики елементи в $A[\ell+1..t]$ са по-големи от q. Сега е ясно, че ако $s+k-1 = \ell$, то $A[\ell] = q$ е наистина k-ти по големина в $A[s..t]$. В противен случай, ако $s+k-1 < \ell$, то k-тият по големина в оригиналния масив е k-тият по големина измежду по-малките от q елементи, тоест в $A[s..\ell-1]$. Ако пък $s+k-1 > \ell$, то k-тият елемент в ориганлния сегмент $A[s..t]$ е $k-(\ell-s)$ след по-големите от q елементи.

Сега да анализираме времевата сложност. Нека $T(m+1) = \max_{A,t-s=m} \mathbb{E}Time(QuickSelectK(A,s,t,k))$ и за $x \in \mathbb{R}^+$, нека:

$$T'(x) = \max\{T(m) \,|\, m \leq x\}.$$

Тогава е ясно, че $T'(x)$ е монотонно растяща.

Нека за $j \in [s..t]$, с r_j означим реда на j в сортирания масив $A[s..t]$. От предположението, че j на ред @1 се избира равномерно и независимо. Тъй като елементите на $A[1..n]$ са различни, то рангът r на $A[s+j]$ се определя еднозначно от j. Следователно вероятността избраният елемент $q = A[j]$ на ред @2 да има ранг $r \leq m+1$ е равна и е равна на $1/(m+1)$. Нека $\mathcal{E}_r$ е събитието $q = A[s+j]$ да има ранг r. Нека:

$$\mathcal{E}_{2/3} = \{j \,|\, (m+1)/3 \leq j \leq 2(m+1)/3\}.$$

Тогава вероятността $\mathbb{P}(\mathcal{E}_{2/3}) \geq 1/3$ и за всяко $r \in \mathcal{E}_{2/3}$, рекурсивното извикване е за интервал $I' = [s..s+r-1]$ или $I'' = [s+r+1..t]$. За $r \in \mathcal{E}_{2/3}$ дължината на интервалите I' и I'' е по-малка или равна от $2(m+1)/3$. Следователно:

$$Time(m) = \mathbb{P}(\mathcal{E}_{2/3})\mathbb{E}[Time(QuickSelectK(A,s,t,k))|\mathcal{E}_{2/3}] + \mathbb{P}(\mathcal{E}^c_{2/3})\mathbb{E}[Time(m)|\mathcal{E}^c_{2/3}]$$

Тъй като когато настъпи $\mathcal{E}_{2/3}$ се изпълняват не повече $m+1$ операции в $Rearrange$, а рекурсивното извикване е към масив с дължина не по-голяма от $\mathbb{E}[Time(m)|\mathcal{E}^c_{2/3}] \leq m+1+\max_{s\leq m} \mathbb{E}Time(s) = m+1+T'(m)$ и получаваме:

$$\begin{aligned} Time(m) &\leq \mathbb{P}(\mathcal{E}_{2/3})(m+1+T'(2/3m)) + (1-\mathbb{P}(\mathcal{E}_{2/3})(m+1+T'(m)) \\ &= \mathbb{P}(\mathcal{E}_{2/3})T'(2/3m) + (1-\mathbb{P}(\mathcal{E}_{2/3})T'(m) + (m+1) \\ (\text{монотонност на } T' \text{ и } \mathbb{P}(\mathcal{E}_{2/3}) \geq 1/3) \quad &\leq \frac{1}{3}T'(2/3m) + 2/3T'(m) + (m+1). \end{aligned}$$

Тъй като това е вярно за всяко m, а T' е монотонно растяща, то:

$$T'(x) = \max_{s \leq x} Time(s) \leq \frac{1}{3}T'(2/3x) + 2/3T'(x) + (x+1).$$

Оттук вече получаваме, че:

$$\begin{gathered} T'(x) \leq T'(2/3x) + 3(x+1) \text{ и въобще за } i \geq 0 \\ T'((2/3)^i x) \leq T'((2/3)^{i+1}x) + 3((2/3)^i x + 1). \end{gathered}$$

Нека j е най-малкото естествено число, за което $(2/3)^j x \leq 1$. Тогава като съберем почленно горните неравенства за $0 \leq i < j$ получаваме:

$$T'(x) \leq T'((2/3)^j x) + 3\sum_{i=0}^{j-1}(2/3)^i x + j \leq T'(1) + 3x/(1-2/3) + j = T'(1) + 9x + \log x.$$

Следователно $T(m) \leq T'(m) \leq 9m + T(1) + \log m$. $\square$

QuickSort($A[1..n]$: array of integers; s, t: indices in $[1,n]$)

@1 if $s \geq t$ return
@2 $p' \leftarrow \mathcal{U}(\{1,\ldots,t-s+1\})$
@3 $p \leftarrow p' + s - 1$
@4 $\ell \leftarrow Rearrange(A,s,t,p)$
@5 $QuickSort(A,s,\ell-1)$
@6 $QuickSort(A,\ell+1,t)$

Теорема 0.1. *При всеки вход от масив $A[1..n]$ от числа и индекси $1 \leq s \leq t \leq n$:*

1. *$QuickSort(A, s, t)$ сортира във възходящ ред подмасива $A[s..t]$, без да променя $A[1..s-1]$ и $A[t+1..n]$;*
2. *ако елементите на A са два по два различни, то $\mathbb{E}Time(QuickSort(A, s, t)) = O((t-s+1)\log(t-s+1))$.*

Доказателство. Коректността следва непосредствено с индукция по $t - s$, като се вземе предвид, че от свойствата на $Rearrange$, $A[s..\ell-1]$ се състои от елементи не ненадминаващи q, а $A[\ell+1..t]$ от такива, които са по-големи от $q = A[\ell]$.

Да разгледаме времевата сложност на алгоритъма. Както и преди, нека:

$$T(m) = \max_{A, t-s=m} \mathbb{E}Time(QuickSort(A, s, t)).$$

Както и в предишното доказателство нека $\mathcal{E}_r$ е събитието, при което $A[s+j]$ има ранг r в $A[s..t]$. Тъй като елементите на $A[s..t]$ са различни, то $\mathcal{E}_r$ настъпва точно когато $\ell = s + r - 1$. Следователно:

$$\begin{aligned}
\mathbb{E}[Time(QuickSort(A, s, t)) \,|\, \mathcal{E}_r] &= m + 1 + \mathbb{E}Time(QuickSort(A, s, s+r-2)) + \mathbb{E}Time(QuickSort(A, s+r, t)) \\
&\leq m + 1 + T(r-2) + T(m-r).
\end{aligned}$$

Тъй като всяко $\mathcal{E}_r$ се случва с вероятност $1/(m+1)$, то:

$$\begin{aligned}
\mathbb{E}[Time(QuickSort(A, s, t))] &= \sum_{r=1}^{m+1} \mathbb{P}(\mathcal{E}_r)\mathbb{E}[Time(QuickSort(A, s, t)) \,|\, \mathcal{E}_r] \\
&\leq \frac{1}{m+1} \sum_{1}^{m+1} (m + 1 + T(r-2) + T(m-r)) \\
&= m + 1 + \frac{2T(-1) + 2T(0)}{m+1} + \frac{2}{m+1} \sum_{r=1}^{m-1} T(r) \\
(T(0) = T(-1) = 1) \quad &\leq m + 5 + \frac{2}{m+1} \sum_{r=1}^{m-1} T(r).
\end{aligned}$$

Да положим $S(0) = S(-1) = 1$ и:

$$S(m) = m + 5 + \frac{2}{m+1} \sum_{r=1}^{m-1} S(r).$$

Тогава по индукция получаваме, че $S(m) \geq T(m)$ за всяко m. Сега:

$$\begin{aligned}
(m+1)S(m) &= (m+1)(m+5) + 2\sum_{r=1}^{m-1} S(r) \text{ и като запишем условието за } m+1 \\
(m+2)S(m+1) &= (m+2)(m+6) + 2\sum_{r=1}^{m} S(r) \text{ и извадим почленно получаваме} \\
(m+2)S(m+1) &= (m+1)S(m) + 2S(m) + 2m + 7 \\
&= (m+3)S(m) + 2m + 7.
\end{aligned}$$

Оттук:

$$\begin{aligned}
\frac{S(m+1)}{m+3} &= \frac{S(m)}{m+2} + \frac{(2m+7)}{(m+2)(m+3)} = \frac{S(m)}{m+2} + \frac{3}{m+2} - \frac{1}{m+3} \\
(\text{по индукция }) \quad &= \frac{S(0)}{2} + \sum_{j=0}^{m} \left(\frac{3}{j+2} - \frac{1}{j+3} \right) \\
&= \frac{1}{2} + 2\sum_{j=0}^{m} \frac{1}{j+2} + \frac{1}{2} - \frac{1}{m+3} \\
&= 1 + 2\sum_{j=2}^{m+2} \frac{1}{j} \leq 1 + 2\sum_{j=1}^{m+1} \int_{j}^{j+1} \frac{1}{j+1} dx \\
&\leq 1 + 2\sum_{j=1}^{m+1} \int_{j}^{j+1} \frac{1}{x} dx \\
&= 1 + 2\int_{1}^{m+2} \frac{1}{x} dx = 1 + 2\log(m+2).
\end{aligned}$$

Следователно $S(m+1) \leq (m+3)(1+2\log(m+2))$. Оттук $T(m) \leq S(m) \leq (m+2)(1+2\log(m+1)) \in O(m \log m)$. $\square$

Задача 0.1. *Да се модифицира Rearrange и да се направи необходимият анализ на QuickSelectK и QuickSort, така че резлутатите за очакваното време на двете функции да е в сила за произволни масиви от числа, а не задължително от такива, в които няма повтарящи се елементи.*