Зад. 1 Решете следните рекурентни уравнения.

$$A(n) = 29A\left(\frac{n}{3}\right) + 2\sum_{k=1}^{n}\frac{1}{k^2} \qquad B(n) = 29B\left(\frac{n}{3}\right) + 12n + \sqrt{n} \qquad C(n) = C(n-1) + \frac{n}{(n-1)(n+1)}$$

$$D(n) = 29D\left(\frac{n}{3}\right) + \left(\sum_{k=1}^{n}\frac{1}{k}\right)^4 \qquad E(n) = 29E\left(\frac{n}{3}\right) + 2\sum_{k=1}^{n}k^2 \qquad F(n) = 29F\left(\frac{n}{3}\right) + n^{\sqrt{n}} + (\sqrt{n})^n$$

$$G(n) = G\left(\sqrt{n}\right) + n \qquad H(n) = 29H\left(\frac{n}{3}\right) + \binom{2n}{2} \qquad K(n) = 8K(n-1) - K(n-2) + 2n2^{2n} + 3n2^{3n}$$

Решение: Повечето от тези уравнения се решават с Мастър Теоремата (МТ), като $a = 29$ и $b = 3$. В тези решения разглеждаме $\log_3 29$, което е приблизително 3.065044751. Точната стойност е без значение. Важното е, че $\log_3 27 < \log_3 29 < \log_3 81$, така че $3 < \log_3 29 < 4$.

Да намерим $A(n)$. Нехомогенната част е $2\sum_{k=1}^{n}\frac{1}{k^2}$. Но това е ограничено от константа. Тогава за нехомогенната част $f(n)$ е изпълнено $f(n) = \Theta(1)$, така че $f(n) = O(n^{\log_3 29}/n^{\epsilon})$ за някое положително ϵ. Според случай 1 на МТ, $A(n) \asymp n^{\log_3 29}$.

Да намерим $B(n)$. Нехомогенната част е $12n + \sqrt{n}$. Тогава за нехомогенната част $f(n)$ е изпълнено $f(n) = \Theta(n)$, така че $f(n) = O(n^{\log_3 29}/n^{\epsilon})$ за някое положително ϵ. Според случай 1 на МТ, $B(n) \asymp n^{\log_3 29}$.

Да намерим $C(n)$. Методът с характеристичното уравнение не е приложим, понеже нехомогенната част не е от правилния вид. Ще го решим с развиване.

$$C(n) = C(n-1) + \frac{n}{(n-1)(n+1)}$$
$$C(n) = C(n-2) + \frac{n-1}{(n-2)(n)} + \frac{n}{(n-1)(n+1)}$$
$$C(n) = C(n-3) + \frac{n-2}{(n-3)(n-1)} + \frac{n}{(n-2)(n)} + \frac{n}{(n-1)(n+1)}$$
$$\cdots$$
$$C(n) = C(1) + \frac{2}{(1)(3)} + \frac{3}{(2)(4)} + \cdots + \frac{n-2}{(n-3)(n-1)} + \frac{n-1}{(n-2)(n)} + \frac{n}{(n-1)(n+1)}$$

$C(1) = \Theta(1)$, така че го игнорираме. Асимптотиката се определя от останалите събираеми.

$$C(n) \asymp \frac{2}{(1)(3)} + \frac{3}{(2)(4)} + \cdots + \frac{n-2}{(n-3)(n-1)} + \frac{n-1}{(n-2)(n)} + \frac{n}{(n-1)(n+1)} = \sum_{k=2}^{n}\frac{k}{(k-1)(k+1)}$$

Да разгледаме общия член на сумата.

$$\frac{k}{(k-1)(k+1)} = \frac{k-1+1}{(k-1)(k+1)} = \frac{k-1}{(k-1)(k+1)} + \frac{1}{(k-1)(k+1)} = \frac{1}{k+1} + \frac{1}{(k-1)(k+1)}$$

Тогава

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

Съгласно изучаваното на лекции, $\sum_{k=2}^{n}\frac{1}{k+1} \asymp \lg n$. От друга страна, $\sum_{k=2}^{n}\frac{1}{(k-1)(k+1)} = \Theta(1)$, понеже редът $\sum_{k=2}^{\infty}\frac{1}{(k-1)(k+1)}$ е сходящ. Следователно, $C(n) \asymp \lg n$.

Да намерим $D(n)$. Нехомогенната част е $\left(\sum_{k=1}^{n} \frac{1}{k}\right)^4$. От лекции знаем, че $\sum_{k=1}^{n} \frac{1}{k} \asymp \lg n$. Тогава за нехомогенната част $f(n)$ е изпълнено $f(n) \asymp (\lg n)^4$, така че $f(n) = O(n^{\log_3 29}/n^{\epsilon})$ за някое положително ϵ. Според случай 1 на МТ, $D(n) \asymp n^{\log_3 29}$.

Да намерим $E(n)$. Нехомогенната част е $2\sum_{k=1}^{n} k^2$. Известно е, че $\sum_{k=1}^{n} k^2 \asymp n^3$; това може да се покаже тривиално чрез метода с характеристичното уравнение. Тогава за нехомогенната част $f(n)$ е изпълнено $f(n) \asymp n^3$, така че $f(n) = O(n^{\log_3 29}/n^{\epsilon})$ за някое положително ϵ, предвид факта, че $3 < \log_3 29$. Според случай 1 на МТ, $E(n) \asymp n^{\log_3 29}$.

Да намерим $F(n)$. Нехомогенната част е $f(n) = n^{\sqrt{n}} + (\sqrt{n})^n$. Твърдим, че $n^{\sqrt{n}} \prec (\sqrt{n})^n$. Наистина, ако логаритмуваме двете функции от сумата, първата е $\sqrt{n}\lg n$, а втората е $n \cdot \frac{1}{2}\lg n$. Тъй като $\sqrt{n}\lg n \prec \frac{n}{2}\lg n$, в сила е $n^{\sqrt{n}} \prec (\sqrt{n})^n$ съгласно свойствата на асимптотичните сравнения на образи на функции след логаритмуване, изучавани на лекции. Тогава за нехомогенната част е изпълнено $f(n) \asymp (\sqrt{n})^n$ и имаме право да разглеждаме този по-прост вид. Очевидно нехомогенната част е суперполиномиална, така че, ако МТ е приложима, то случай 3 ще е приложим. Остава да се убедим, че условието за регулярност е изпълнено. Условието за регулярност в този случай гласи "съществува константа c, такава че $0 < c < 1$ и за всички достатъчно големи n е изпълнено $29(\sqrt{n/3})^{n/3} < c(\sqrt{n})^n$ ". Очевидно такава c съществува, така че съгласно случай 3 на МТ, $F(n) \asymp (\sqrt{n})^n$.

Да намерим $G(n)$. С развиване получаваме

$$G(n) = G\left(\sqrt{n}\right) + n$$
$$G(n) = G\left(\sqrt[4]{n}\right) + \sqrt{n} + n$$
$$G(n) = G\left(\sqrt[8]{n}\right) + \sqrt[4]{n} + \sqrt{n} + n$$
$$\cdots$$
$$G(n) = \Theta(1) + \cdots + \sqrt[8]{n} + \sqrt[4]{n} + \sqrt{n} + n$$

Съгласно изучаваното на лекции, тази сума има $\Theta(\lg\lg n)$ събираеми. Но тогава броят на събираемите без n е $O(\lg\lg n)$, а най-големото от тях е $\sqrt{n}$, така че, изключвайки n, сумата е $O(\sqrt{n}\lg\lg n)$. Тъй като $\sqrt{n}\lg\lg n \prec n$, събираемото n доминира над сумата от останалите събираеми и дава асимптотичното нарастване. Накратко, $G(n) \asymp n$.

Да намерим $H(n)$. Нехомогенната част е $\binom{2n}{2}$. Но $\binom{2n}{2} = 2n(2n-1)/2 \asymp n^2$. Тогава за нехомогенната част $f(n)$ е изпълнено $f(n) = \Theta(n^2)$, така че $f(n) = O(n^{\log_3 29}/n^{\epsilon})$ за някое положително ϵ. Според случай 1 на МТ, $B(n) \asymp n^{\log_3 29}$.

Да намерим $K(n)$. Ще използваме метода с характеристичното уравнение, като първо препишем нехомогенната част по такъв начин, че да има правилната форма.

$$K(n) = 8K(n-1) - K(n-2) + 2n4^n + 3n8^n$$

Характеристичното уравнение е

$$x^2 - 8x + 1 = 0$$

с корени $x_1 = 4 + \sqrt{15}$ и $x_2 = 4 - \sqrt{15}$. x_1 е приблизително 7.872983346, а x_2 е приблизително 0.127016654. Точните стойности нямат значение. Важното е, че са положителни, $x_1 > x_2$ и $x_1 < 8$. В общото решение от нехомогенната част влизат две четворки и две осмици. Тъй като $x_1 < 8$, в сила е $K(n) \asymp n8^n$.

Зад. 2 Разгледайте следния алгоритъм.

```
AlgX(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])
```

1 т. Какво прави алгоритъмът?

23 т. Докажете това формално и прецизно.

Решение: Този алгоритъм сортира и е известен като BUBBLE SORT.

Лема По отношение на едно фиксирано изпълнение на външния цикъл, ако в началото на това изпълнение е вярно, че подмасивът $A[n+2-i,\ldots,n]$ се състои от $i-1$ най-големи елемента на входа в сортиран вид, то при приключването на това изпълнение е вярно, че подмасивът $A[n+1-i,\ldots,n]$ се състои от i най-големи елемента на входа в сортиран вид.

Доказателство Разглеждаме кое да е изпълнение на външния цикъл. Инвариант за вътрешния цикъл е следното твърдение: при всяко достигане на ред 2, i-ият най-голям елемент на входа е максималният елемент в $A[j,\ldots,n+1-i]$.

/* i-ият най-голям елемент означава следното. Ако $i = 1$, става дума на максималния елемент. Ако $i = 2$, за втория максимален. И така нататък. Ако $i = n$, става дума за минималния. За да е сортиран масивът, трябва i-ият по големина да се намира на позиция $n+1-i$. */

Базата е $j = 1$. Твърдението е тривиално вярно при направеното допускане, че $A[n+2-i,\ldots,n]$ се състои от $i-1$ най-големи елемента. Тогава i-ият по големина не може да е в $A[n+2-i,\ldots,n]$ и остава да е някъде в $A[1,\ldots,n+1-i]$, което е $A[j,\ldots,n+1-i]$ при $j = 1$. Това, че този i-и по големина елемент се явява максимумът на $A[j,\ldots,n+1-i]$, следва веднага от допускането, че $A[n+2-i,\ldots,n]$ се състои от $i-1$ най-големи елемента на входа.

Поддръжка. Нека твърдението е вярно за някое достигане на ред 2, което не е последното. Следните възможности са взаимно изключващи се и изчерпателни.

- $A[j] > A[j+1]$. Алгоритъмът разменя $A[j]$ с $A[j+1]$. След това разменяне на елементи е вярно, че максимумът на $A[j,\ldots,n+1-i]$ е максимумът на $A[j+1,\ldots,n+1-i]$. Следователно, i-ият по големина от входа гарантирано е в $A[j+1,\ldots,n+1-i]$. След инкрементирането на j отново е изпълнено, че i-ият най-голям елемент на входа е максималният елемент в $A[j,\ldots,n+1-i]$.

- $A[j] \le A[j+1]$. Алгоритъмът не разменя елементи. Очевидно е вярно, че максимумът на $A[j,\ldots,n+1-i]$ е максимумът на $A[j+1,\ldots,n+1-i]$. Следователно, i-ият по големина от входа гарантирано е в $A[j+1,\ldots,n+1-i]$. След инкрементирането на j отново е изпълнено, че i-ият най-голям елемент на входа е максималният елемент в $A[j,\ldots,n+1-i]$.

Терминация. При последното достигане на ред 2 е в сила $j = n+1-i$. Заместваме в доказания инвариант и получаваме "i-ият най-голям елемент на входа е максималният елемент в $A[n+1-i,\ldots,n+1-i]$". С други думи, i-ият най-голям елемент във входа е именно $A[n+1-i]$. Заедно с допускането, че $A[n+2-i,\ldots,n]$ се състои от $i-1$ най-големи елемента на входа в сортиран вид, това влече, че в този момент е вярно, че подмасивът $A[n+1-i,\ldots,n]$ се състои от i най-големи елемента на входа в сортиран вид.

Теорема AlgX е сортиращ алгоритъм.

Доказателство Следното твърдение е инвариант за външния цикъл: при всяко достигане на ред 1, подмасивът $A[n+2-i,\ldots,n]$ се състои от $i-1$ най-големи елемента на входа в сортиран вид.

Базата е $i = 1$. Твърдението става "подмасивът $A[n+1,\ldots,n]$ се състои от 0 най-големи елемента на входа в сортиран вид", което е вярно в празния смисъл.

Поддръжка. Нека твърдение е в сила за някое достигане на ред 1, което не е последното. Съгласно лемата, след изпълнението на вътрешния цикъл е вярно, че $A[n+1-i,\ldots,n]$ се състои от i най-големи елемента на входа в сортиран вид. После i бива инкрементирано. Спрямо новото i, твърдението става "$A[n+2-i,\ldots,n]$ се състои от $i-1$ най-големи елемента на входа в сортиран вид".

Терминация. При последното достигане на ред 1 е вярно, че $i = n+1$. Заместваме в инварианта и получаваме "подмасивът $A[1,\ldots,n]$ се състои от n най-големи елемента на входа в сортиран вид.", което и трябваше да докажем.

Зад. 3 В тази задача става дума за функции, изучавани на лекции.

2 т. Напишете псевдокод за функцията HEAPIFY.

6 т. Накратко обосновете коректността ѝ. Достатъчно е да формулирате смислен инвариант.

2 т. Каква е сложността по време на HEAPIFY? Обосновете отговорите си.

1 т. Напишете псевдокод за функцията Build Heap.
6 т. Накратко обосновете коректността ѝ. Достатъчно е да формулирате смислен инвариант.
3 т. Каква е сложността по време на Build Heap? Обосновете отговорите си.

Решение: *Това е изучавано на лекции*

1 т. **Зад. 4** Каква е средната сложност по време на алгоритъма Quicksort?
19 т. Обосновете подробно отговорите си.

Решение: *Това е изучавано на лекции*

Зад. 5 Представете си игра между двама опоненти, да ги наречем X и Y, в която Y си намисля число a от $\{1, 2, \ldots, n\}$, а X трябва да познае числото. n е дадено и е известно поначало и на двамата играчи. Единствените въпроси, които X може да задава на Y, са от вида "Дали a е по-малко от k?", за някакво естествено k. Може да мислите, че X подава стойност k на Y, а Y връща ДА или НЕ за k, после X подава друга стойност k' на Y, а Y връща ДА или НЕ за k', и така нататък. Какви стойности подадва X и в какъв ред е изцяло избор на X. Y само отговаря. Целта на X е да изчисли a с минимален брой въпроси. Целта на Y е да накара X да зададе максимален брой въпроси. Мислете за Y като за опонента от лекциите за долни граници: Y има неограничена изчислителна мощ и е злонамерен.

Намерете и обосновете колкото можете по-висока асимптотична долна граница за броя на въпросите, които X задава в най-лошия случай, за да реши задачата, имайки предвид, че Y е опонент. Иска се да използвате аргументация с противник: опишете стратегията на Y.

Решение: Ще докажем долна граница $\Omega(\lg n)$ за броя на въпросите чрез аргументация с противник. Нека S е множеството от числата, в които може да е a по отношение на въпросите, зададени досега, и получените отговори. В началото $S = \{1, \ldots, n\}$, така че $|S| = n$. След всеки въпрос на X, който съдържа число $k \in \{1 \ldots n\}$, противникът Y преценява кой отговор—ДА или НЕ—води до по-голямо S. По този начин S може да намалее наполовина, но не повече. За да може X да отговори, трябва $|S| = 1$.

Ако кажем, че S_0 е началното S, а S_i е полученото S след i-ия въпрос, в сила е следната зависимост

$$|S_0| = n$$
$$|S_{i+1}| \geq \left\lceil \frac{|S_i|}{2} \right\rceil, \text{ ако } i \geq 1$$

При това положение, минималната стойност за i, такава че $|S_i| = 1$, е $i = \lceil \log_2 n \rceil$. Тъй като ни интересува само асимптотична долна граница, казваме, че тя е $\Omega(\lg n)$.

Зад. 6 Представете си игра между X и Y, в която Y разполага с n естествени числа $a_1, \ldots, a_n$. Известно е, че съществуват $\ell, m \in \mathbb{N}^+$, такива че $\ell < m$ и $\forall i \in \{1, \ldots, n\} : a_i \in \{\ell, \ldots, m\}$. X знае ℓ и m, но не знае числата $a_1, \ldots, a_n$. Играта се състои в това, X да задава въпроси на Y от вида "Колко числа измежду $a_1, \ldots, a_n$ са по-големи или равни на a и по-малки или равни на b?". Може да мислите, че въпросите на X се състоят само от наредени двойки (a, b), където $\ell \leq a < b \leq m$. Y трябва да отговори коректно на всяко запитване на X.

Тази задача не е за долни граници и Y не е противник! Y сега казва истината. От вас се иска да намерите алгоритъм за Y, който по наредена двойка (a, b) отговаря във време $\Theta(1)$ на въпроса на X. На Y е разрешено да извърши предварителна обработка на данните във време $O(n + (m - \ell))$.
15 т. Опишете предварителната обработка, която Y извършва.
10 т. Опишете алгоритъма, по който Y отговаря на X в константно време.

Решение: Y прави предварителна обработка, подобна на тази от Counting Sort. Тъй като възможните стойности за a_i не започват от единица, а от някакво ℓ, има смисъл работният масив C да е с граници $C[\ell - 1, \ldots, m]$. Y прави следната предварителна обработка.

```
Alg1(A[1, ..., n]: цели положителни; ℓ, m: цели положителни)
1  (* ℓ и m са такива, че ℓ < m и ℓ ≤ A[i] ≤ m за всички i *)
2  for i ← ℓ − 1 to m
```

```
3       C[i] ← 0
4   for i ← 1 to n
5       C[A[i]] ← C[A[i]] + 1
6   for i ← ℓ to m
7       C[i] ← C[i] + C[i − 1]
```

Това, че алгоритъмът работи във време $O(n + (m - \ell))$ се показва по същия начин, по който се показва, че сложността по време на Counting Sort е $O(n + k)$. В случая $m - \ell$ играе ролята на k.

От лекции знаем, че след третия цикъл за всяко j, такова че $\ell \leq j \leq m$, $C[j]$ съдържа броят на елементите в масива A, които са по-малки или равни на j. Тогава $C[b] - C[a-1]$ е броят на елементите в $w \in A$, за които

$$w \leq b \wedge \neg(w \leq a - 1) \;\leftrightarrow\; a - 1 < w \leq b \;\leftrightarrow\; a \leq w \leq b$$

Така че при запитване, съдържащо (a, b), Y връща $C[b] - C[a-1]$. Това се изчислява във време $\Theta(1)$, понеже Y разполага с C.