Решения на контролното по ДАА от 19.04.2015 г.

Зад. 1 Докажете или опровергайте, че ако $f(n) = O(g(n))$ и $f(n) \neq \Theta(g(n))$, то $f(n) = o(g(n))$.

Решение: Твърдението не е вярно. Нека $n \in \mathbb{N}^+$. Контрапример са $g(n) = n$ и

$$f(n) = \begin{cases} n, & \text{ако } n \text{ е четно} \\ 1, & \text{в противен случай} \end{cases}$$

Да се убедим, че тези две функции са контрапример.

- Ако вземем $c = 1$ и $n_0 = 1$ в дефиницията на O-голямо, очевидно е изпълнено

  $$\forall n \geq n_0 : f(n) \leq cg(n)$$

  понеже това е същото като

  $$\forall n \geq 1 : f(n) \leq g(n)$$

  Следователно е изпълнено $f(n) = O(g(n))$.

- Да допуснем, че за някои положителни константи c_1, c_2 и n_0 е изпълнено $c_1 g(n) \leq f(n) \leq c_2 g(n)$, за всички $n \geq n_0$. Тогава в частност е вярно, че

  $$\forall n \geq n_0 : c_1 g(n) \leq f(n)$$

  Нека $\widetilde{\mathbb{N}_e}$ са нечетните числа, по-големи от n_0. Съгласно допускането ни,

  $$\forall n \in \widetilde{\mathbb{N}_e} : c_1 g(n) \leq f(n)$$

  Но при нечетните аргументи, $f(n) = 1$. Извеждаме съществуването на константа c_1, такава че

  $$\forall n \in \widetilde{\mathbb{N}_e} : c_1 n \leq 1 \qquad \leftrightarrow \qquad \forall n \in \widetilde{\mathbb{N}_e} : n \leq \frac{1}{c_1}$$

  Очевидно е, че такава константа няма. Тогава допускането е невярно. Следователно, $f(n) \neq \Theta(g(n))$.

- За да се убедим, че дадените $f(n)$ и $g(n)$ наистина са контрапример, трябва да покажем, че $f(n) \neq o(g(n))$. Да допуснем, че $f(n) = o(g(n))$. Тогава за всяка положителна константа c съществува стойност n_0, такава че $f(n) < cg(n)$ за всяко $n \geq n_0$. Нека $\tilde{c}$ е произволно положително число **без ограничения** и $\tilde{n}_0$ е стойност на аргумента n, такава че $f(n) < \tilde{c}g(n)$ за всяко $n \geq \tilde{n}_0$. Нека $\widetilde{\mathbb{N}_o}$ са четните числа, по-големи от $\tilde{n}_0$. Съгласно допускането ни,

  $$\forall n \in \widetilde{\mathbb{N}_o} : f(n) < \tilde{c}g(n)$$

  Но при четните аргументи, $f(n) = n$. Изведохме, че за произволно положително число **без ограничения** $\tilde{c}$ е вярно, че

  $$\forall n \in \widetilde{\mathbb{N}_o} : n < \tilde{c}n \qquad \leftrightarrow \qquad \forall n \in \widetilde{\mathbb{N}_o} : 1 < \tilde{c}$$

  Това очевидно не е вярно, понеже "произволно положително число **без ограничения**" имплицира, че числото може да е по-малко от единица. □

Зад. 2 Подредете по асимптотично нарастване следните десет функции. Обосновете отговорите си кратко. Напишете в явен вид самата подредба.

$f_1 = (\lg n)^{\lg n}$, $\quad f_2 = n^{\lg n!}$, $\quad f_3 = n^3 + 3n^2$, $\quad f_4 = 1 + \binom{\lg n}{\frac{1}{2}\lg n}$, $\quad f_5 = n^2$

$f_6 = \sum_{k=1}^{n} \frac{1}{k}$, $\quad f_7 = \sum_{k=1}^{n} \frac{1}{k^2}$, $\quad f_8 = n^{\lg \lg n}$, $\quad f_9 = 1 + \binom{\lg n}{n}$, $\quad f_{10} = \lg \lg n$

Заради функции f_4 и f_9 можете да допуснете, че n е точна степен на двойката.

Решение:

(i) Ще докажем, че $f_2 \succ f_1$. Ако логаритмуваме двете функции, получаваме $\lg(f_1) = (\lg n) \lg\lg n$ и $\lg(f_2) = (\lg n!) \lg n$. От упражнения е известно, че $\lg\lg n \prec \lg n$, така че $\lg(f_1) \prec (\lg n)^2$. Изведохме, че $\lg(f_1)$ расте по-бавно, в асимптотичния смисъл, от полилогаритмична функция.

От друга страна, $\lg(f_2) \succ (\lg n!)$. Известно е, че $\lg(n!) \succ n$. Изведохме, че $\lg(f_2) \succ n$, тоест $\lg(f_2)$ расте по-бързо, в асимптотичния смисъл, от полиномиална функция.

От упражнения е известно, че всяка полиномиална функция расте по-бързо от всяка полилогаритмична. Извеждаме, че $\lg(f_2) \succ \lg(f_1)$. Съгласно изучаваното на упражнения, това влече, че $f_2 \succ f_1$.

(ii) Ще докажем, че $f_1 \asymp f_8$. Но всъщност това е една и съща функция, което се вижда веднага след логаритмуване на двата израза:

$$\begin{aligned} \lg\left((\lg n)^{\lg n}\right) &= (\lg n) \lg\lg n \\ \lg\left(n^{\lg\lg n}\right) &= (\lg n) \lg\lg n \end{aligned}$$

(iii) Ще докажем, че $f_8 \succ f_3$. Веднага се вижда, че

$$\lim_{n\to\infty} \frac{n^{\lg\lg n}}{n^3 + 3n^2} = \lim_{n\to\infty} \frac{n^{(\lg\lg n)-3}}{1 + 3\frac{1}{n}} = \infty$$

(iv) Това, че $f_3 \succ f_5$, тоест $n^3 + 3n^2 \succ n^2$, е очевидно.

(v) Ще докажем, че $f_5 \succ f_4$. Да разгледаме

$$f_4 = 1 + \binom{\lg n}{\frac{1}{2}\lg n}$$

Сега да разгледаме, за четни стойности на аргумента m,

$$\binom{m}{\frac{m}{2}}$$

Прилагайки апроксимацията на Стирлинг и това, което знаем за биномния коефициент, получаваме

$$\binom{m}{\frac{m}{2}} = \frac{\sqrt{2\pi m}\,\frac{m^m}{e^m}}{\left(\sqrt{2\pi\frac{n}{2}}\,\frac{\frac{m}{2}^{\frac{m}{2}}}{e^{\frac{m}{2}}}\right)^2} = \sqrt{2\pi}\sqrt{m}\,\frac{m^m}{e^m} \times \frac{e^m}{\pi\, m\,\frac{m^m}{2^m}} = \sqrt{\frac{2}{\pi}} \times \frac{1}{\sqrt{m}} \times 2^m \asymp \frac{2^m}{\sqrt{m}}$$

Замествайки m с $\lg n$, получаваме

$$\binom{\lg n}{\frac{1}{2}\lg n} \asymp \frac{n}{\sqrt{\lg n}}$$

Тогава

$$f_4 \asymp \frac{n}{\sqrt{\lg n}}$$

и е очевидно, че $f_5 \succ f_4$.

(vi) Ще докажем, че $f_4 \succ f_6$. Но f_6 е n-тата парциална сума на хармоничния ред. За нея е доказано на лекции, че като асимптотично нарастване е еквивалентна на $\lg n$. Тъй като f_4 расте по-бързо, в асимптотичния смисъл, от някоя полиномиална функция, примерно $n^{\frac{1}{2}}$ и всяка полиномиална функция расте по-бързо, в асимптотичния смисъл, от всяка полилогаритмична функция, твърдението следва веднага.

(vii) Ще докажем, че $f_6 \succ f_{10}$. Вече показахме, че $f_6 \asymp \lg n$. Известно е от упражнения, че $\lg n \succ \lg \lg n$, и от това и дефиницията на f_{10} твърдението следва веднага.

(viii) Ще докажем, че $f_{10} \succ f_7$. За целта ще покажем, че $f_7 = \Theta(1)$. Но това следва веднага от факта, че редът $\sum_{k=1}^{n} \frac{1}{k^2}$ е сходящ. Алтернативно, можем да използваме следното разсъждение

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

Щом f_{10} расте неограничено f_7 е ограничена, твърдението следва веднага.

(ix) Ще докажем, че $f_7 \asymp f_9$. Вече доказахме, че $f_7 = \Theta(1)$. Ще докажем, че $f_9 = \Theta(1)$. Да разгледаме f_9:

$$f_9 = 1 + \binom{\lg n}{n}$$

Известно е, че биномният коефициент е нула, ако долният индекс е по-голям от горния. Но n е по-голямо от $\lg n$ за всички достатъчно големи стойности на n. Тогава биномният коефициент в дефиницията на f_9 е нула, така че $f_9 \asymp 1$.

От **(i)**–**(ix)** следва наредбата:

$$f_2 \succ f_1 \asymp f_8 \succ f_3 \succ f_5 \succ f_4 \succ f_6 \succ f_{10} \succ f_7 \asymp f_9$$

□

Зад. 3 Даден е масив $A[1, \ldots, n]$ от цели различни числа. *Склон* в $A[1, \ldots, n]$ ще наричаме всеки максимален по включване подмасив $A[i, \ldots, j]$, такъв че $i \leq j$ и $A[k] < A[k+1]$ за $i \leq k < j$.

2 т. • Предложете алгоритъм със сложност по време $O(n)$ и сложност по памет $\Theta(1)$, който изчислява броя на склоновете.

18 т. • Докажете формално коректността на Вашия алгоритъм.

Решение Следният алгоритъм решава задачата:

```
Slopes(A[1,...,n] : integer;   i ≠ j → A[i] ≠ A[j])
1  if n = 0
2      return 0
3  count ← 0
4  for i ← 2 to n
5      if A[i − 1] > A[i]
6          count ← count + 1
7  return count
```

Сложността му по време е $\Theta(n)$, понеже цикълът се изпълнява $\Theta(n)$ пъти и всяко негово изпълнение отнема $\Theta(1)$ време. Сложността по памет очевидно е $\Theta(1)$, тъй като алгоритъмът ползва само две променливи от целочислен тип.

Сега ще докажем коректността му. Ако $n = 0$, в масива няма склонове (това не е казано експлицитно в дефиницията на "склон", така че може спокойно да пропуснем редове 1 и 2 и да не се занимаваме с възможността $n = 0$) и алгоритъмът наистина връща 0 (ред 2). Нека $n > 0$. Следното твърдение е инварианта на **for**-цикъла.

При всяко достигане на ред 4, променливата count съдържа броя на склоновете в подмасива $A[1, \ldots, i-1]$.

База При първото достигане на ред 4 е изпълнено $i = 2$, така че подмасивът $A[1, \ldots, i-1]$ е $A[1]$. В него очевидно има точно един склон. От друга страна, count има стойност 1 заради присвояването на ред 3. ✓

Запазване Нека твърдението е вярно при някое достигане, което не е последното. Ще разгледаме два случая: $A[i-1] > A[i]$ и $A[i-1] \leq A[i]$.

- Нека $A[i-1] > A[i]$. Тогава се изпълнява присвояването на ред 6 и стойността на count става равна на броя на склоновете в $A[1, \ldots, i-1]$ (заради индуктивното предположение), плюс едно. Но в този случай $A[i-1]$ е най-десният елемент на някой склон и $A[i]$ очевидно е един склон сам по себе си в $A[1, \ldots, i]$. Следователно, променливата count съдържа стойност, равна на броя на склоновете в $A[1, \ldots, i]$. След инкрементирането на i при следващото достигане на ред 4 е вярно, че count съдържа броя на склоновете в подмасива $A[1, \ldots, i-1]$ спрямо новата стойност на i.

- Нека $A[i-1] \leq A[i]$. Присвояването на ред 6 не се изпълнява и стойността на count остава равна на броя на склоновете в $A[1, \ldots, i-1]$ (заради индуктивното предположение). Но в този случай $A[i]$ е най-десен елемент на склона в $A[1, \ldots, i]$, който съдържа $A[i-1]$. С други думи, броят на склоновете в $A[1, \ldots, i-1]$ е равен на броя на склоновете в $A[1, \ldots, i]$. Следователно, променливата count съдържа стойност, равна на броя на склоновете в $A[1, \ldots, i]$. След инкрементирането на i при следващото достигане на ред 4 е вярно, че count съдържа броя на склоновете в подмасива $A[1, \ldots, i-1]$ спрямо новата стойност на i.

Терминация При последното достигане на ред 4 е вярно, че $i = n + 1$. Замествайки тази стойност в инварианатата, получаваме твърдението "променливата count съдържа броя на склоновете в $A[1, \ldots, n]$". Виждаме, че на ред 7 алгоритъмът връща точно тази стойност, която трябва да върне. □

Зад. 4 Дадени са 4 топки. Всяка е надписана с точно едно от A, B, C, D. Известно е, че една от топките тежи 1 килограм, една тежи 2 килограма, една тежи 3 килограма и една тежи 4 килограма. Дадена е везна без теглилки с две блюда, в които може да се слагат топките. Целта е да определим коя от топките колко килограма тежи, използвайки везната, като я използваме колкото е възможно по-малко пъти. Докажете, че броят на измерванията е поне 3.

Решение: Лесно се вижда, че има $4! = 24$ възможности за теглата на четирите топки. Тъй като има три възможни изхода от дадено измерване:

1. лявото блюдо потъва надолу,
2. двете блюда са в равновесие, и
3. дясното блюдо потъва надолу,

дървото на вземане на решение е троично. С две измервания бихме могли да различим най-много $3^2 = 9$ ситуации. А ситуациите са, както казахме, 24. Следователно никаква схема за измерване не могла да използва най-много две измервания в най-лошия случай. С други думи, 3 е долна граница за броя на измерванията. □

Зад. 5 Решете следните рекурентни отношения:

$$T(n) = 2T\left(\frac{n}{3}\right) + 1, \qquad P(n) = \sqrt[3]{5}\,T\left(\frac{n}{\sqrt{5}}\right) + 1,$$

$$Q(n) = 2Q(n-1) + 2^{\frac{n}{2}} + \sqrt[4]{4^n}, \qquad R(n) = nR(n-1) + 1$$

Относно $R(n)$: достатъчно е да го решите чрез развиване.

Решение: $T(n) = 2T\left(\frac{n}{3}\right) + 1$ се решава с Мастър теоремата. Тъй като $1 = n^0$ и $0 < \log_3 2 < 1$, то $1 = O(n^{(\log_3 2) - \epsilon})$ за някое положително ϵ и първият случай на МТ е приложим. Съгласно него, $T(n) = \Theta(n^{\log_3 2})$.

$P(n) = \sqrt[3]{5}T\left(\frac{n}{\sqrt{5}}\right) + 1$ също се решава с МТ. Тъй като $1 = n^0$ и $\log_{\sqrt{5}} \sqrt[3]{5} = \frac{\frac{1}{3}}{\frac{1}{2}} \log_5 5 = \frac{2}{3}$, то $1 = O(n^{\frac{2}{3}-\epsilon})$ за някое положително ϵ и първият случай на МТ е приложим. Съгласно него, $T(n) = \Theta(n^{\frac{2}{3}})$.

$Q(n) = 2Q(n-1) + 2^{\frac{n}{2}} + \sqrt[4]{4^n}$ се решава чрез метода с характеристичното уравнение, ако бъде преобразувано в еквивалентната форма $Q(n) = 2Q(n-1) + (2n^0) \times \sqrt{2}^n$. Характеристичното уравнение има един единствен корен 2. От нехомогенната част идва още един корен $\sqrt{2}$ с кратност 1. Общото решение е $T(n) = A2^n + B\sqrt{2}^n$ за някакви константи A и B. Тогава $T(n) = \Theta(2^n)$.

$R(n)$ ще решим чрез развиване. Нека началното условие е $R(0) = c$ за някоя константа c. Тогава:

$$
\begin{aligned}
R(n) &= nR(n-1) + 1 \\
&= n((n-1)R(n-2) + 1) + 1 \\
&= n(n-1)R(n-2) + n + 1 \\
&= n(n-1)((n-2)R(n-3) + 1) + n + 1 \\
&= n(n-1)(n-2)R(n-3) + n(n-1) + n + 1 \\
&= n(n-1)(n-2)(n-3)R(n-4) + n(n-1)(n-2) + n(n-1) + n + 1 \\
&\dots \\
&= n(n-1)\cdots 1R(0) + n(n-1)\cdots 2 + n(n-1)\cdots 2 + \cdots + n(n-1) + n + 1 \\
&= c \times n! + n! \underbrace{\left(\frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{(n-2)!} + \frac{1}{(n-1)!} + \frac{1}{n!}\right)}_{A}
\end{aligned}
$$

Сумата A е ограничена от константа, защото редът $\sum_{k=1}^{\infty} \frac{1}{k!}$ е сходящ със сума $e-1$. Тогава $R(n) = \Theta(n!)$.
□

Зад. 6 Задачата Търсене в сортиран масив е: при даден сортиран масив $A[1, \dots, n]$ и дадено число k да се отговори дали k се съдържа в $A[1, \dots, n]$ или не. Известно е, че можем да търсим ефикасно в сортиран масив чрез двоично търсене във време $\Theta(\lg n)$ в най-лошия случай.

Докажете, че задачата Търсене в сортиран масив има долна граница $\Omega(\lg n)$, ако търсенето е базирано на директни сравнения. Това означава, че достъпът до елементите на $A[]$ става по един единствен начин: само чрез сравнения от вида $A[i] \stackrel{?}{=} k$.

Решение Без ограничение на общността, нека елементите са различни (понеже търсим долна граница в най-лошия случай, можем да допуснем това; ако изведем някаква долна граница при това предположение, тя е валидна долна граница изобщо).

Лесно можем да посочим $n+1$ "ситуации" по отношение на Търсене в сортиран масив: k може да е равно на някое от числата в масива (n възможности) или не (една възможност). Тъй като дървото на вземането на решение е двоично, то има поне $n+1$ листа, следователно височината му е поне $\log_2 n$, което определя и асимптотична долна граница $\Theta(\lg n)$ за задачата. □