Решения на контролното по ДАА на КН2 от 14.05.2017 г.

Зад. 1 Нека $n \in \mathbb{N}$ и $f(n) \asymp g(n)$. Докажете или опровергайте, че

а) $2f(n) \succeq g(n) + 2$;

б) $2f(n) \succeq g(n+2)$;

в) $2f(n) \succeq g(n+2)$ или $2f(n) \preceq g(n+2)$.

Обосновете отговорите си подробно.

Решение:

а) Твърдението не е вярно. Щеше да е вярно, ако функциите бяха растящи, но такова нещо не е казано. Щом използваме асимптотичните нотации, то налице е имплицитно допускане, че функциите са положителни. Друго имплицитно допускане няма.

Като контрапример да разгледаме $f(n) = g(n) = \frac{1}{n}$. Тези функции са положителни за положителни стойности на n, следователно имаме право да ги разглеждаме. Със сигурност $f(n) \asymp g(n)$ е вярно щом функциите съвпадат. Да допуснем, че $2f(n) \succeq g(n)+2$. По дефиниция това е кратък запис на

$$\exists c' \, \exists n_0' \, \forall n \geq n_0' : c' \frac{1}{n} + 2 \leq 2\frac{1}{n}$$

Но тогава

$$\exists c' \, \exists n_0' \, \forall n \geq n_0' : c' + 2n \leq 2$$

Това е абсурд, имайки предвид, че c' е константа, а n е променлива, която расте неограничено.

б) Твърдението не е вярно. Дори ако разгледаме растящи функции, пак не е вярно. Нека $f(n) = g(n) = 2^{2^n}$. Със сигурност $f(n) \asymp g(n)$ е вярно щом функциите съвпадат.

Но забележете, че $g(n+1) = 2^{2^{n+1}} = (g(n))^2 = (f(n))^2$. Тогава $g(n+2) = (f(n))^4$. Очевидно не е вярно, че $2f(n) \succeq (f(n))^4$ за растяща функция $f(n)$.

в) Твърдението не е вярно. Нека отново $f(n) = g(n)$, дефинирани върху $\mathbb{N}^+$ така

$$f(n) = g(n) = \begin{cases} 2^{2^n}, & \text{ако } n \text{ е четно} \\ 1, & \text{ако } n \bmod 4 = 1 \\ 2^{2^n}, & \text{ако } n \bmod 4 = 3 \end{cases}$$

Както вече видяхме, $2^{2^{n+2}} = \left(2^{2^n}\right)^4$.

От една страна, не е вярно, че $2f(n) \succeq g(n+2)$, защото стойността на функцията върху $n+2$ е четвъртата степен на стойността на функцията върху n. По-подробно казано, за всяка предварително избрана константа c и за всяко n_0 съществува (четно) $n \geq n_0$, такива че $2f(n) < cg(n+2)$. Но това е точно логическото отрицание на условието за $2f(n) \succeq g(n+2)$.

Да разгледаме нечетните числа. Ще разглеждаме двойки последователни нечетни числа, по-малкото от които дава остатък 3 по модул 4, което означава, че по-голямото (тоест, следващото нечетно число) дава остатък 1 по модул 4. Съществуват безброй много такива двойки. Тогава съществуват безброй много двойки n, $n+2$, такива че $f(n) = g(n) = 2^{2^n}$ и $f(n+2) = g(n+2) = 1$. И така, за всяка предварително избрана константа c и за всяко n_0 съществува $n \geq n_0$, такива че $2f(n) > cg(n+2)$. Но това е точно логическото отрицание на условието за $2f(n) \preceq g(n+2)$.

Доказахме, че нито $2f(n) \succeq g(n+2)$, нито $2f(n) \preceq g(n+2)$. □

Зад. 2 Решете следните рекурентни уравнения:

$$T(n) = 3T\left(\frac{n}{3}\right) + 3n, \qquad S(n) = S\left(\frac{n}{2}\right) + S\left(\frac{n}{4}\right) + \cdots + S(1) + n$$
$$Q(n) = 3Q(n-1) + 3n3^n + 3n(3+n)3^n, \qquad R(n) = \sqrt{n}R(n-1) + 1$$

Относно $S(n)$: допуснете, че n е точна степен на двойката. Относно $R(n)$: достатъчно е да го решите чрез развиване, като не е необходимо да сте много прецизни.

Решение Първото уравнение се решава с Мастър теоремата. $a = 3$, $b = 3$, тогава $n^{\log_b a} = n$ и сравняваме n с $3n$. Очевидно $n \asymp 3n$ и съгласно вторият случай на МТ, $T(n) \asymp n \lg n$.

Уравнението за $S(n)$ може да се реши така:

$$S(n) = S\left(\frac{n}{2}\right) + S\left(\frac{n}{4}\right) + S\left(\frac{n}{8}\right) + \cdots + S(1) + n$$
$$S\left(\frac{n}{2}\right) = \qquad S\left(\frac{n}{4}\right) + S\left(\frac{n}{8}\right) + \cdots + S(1) + \frac{n}{2}$$

Изваждаме второто от първото и получаваме

$$S(n) = 2S\left(\frac{n}{2}\right) + \frac{n}{2}$$

Решението е $S(n) \asymp n \lg n$ съгласно втория случай на МТ.

Уравнението за $Q(n)$ се решава тривиално чрез метода с характеристичното уравнение, ако първо препишем даденото уравенение така, че нехомогенната чест да е в правилна форма:

$$Q(n) = 3Q(n-1) + (12n + 3n^2)3^n$$

Мултиммножеството от корените след добавянето на тройките от нехомогенната част е $\{3,3,3,3\}_M$. Тогава $Q(n) \asymp n^3 3^n$.

Ще решим с развиване

$$R(n) = \sqrt{n}R(n-1) + 1$$

Имаме

$$\begin{aligned}
R(n) &= \sqrt{n}R(n-1) + 1 \\
R(n-1) &= \sqrt{n-1}R(n-2) + 1 \\
R(n-2) &= \sqrt{n-2}R(n-3) + 1 \\
&\cdots \\
R(2) &= \sqrt{2}R(1) + 1
\end{aligned}$$

Тук допускаме без ограничение на общността, че неназованото начално условие е $R(1) = c$ за някаква положителна константа c. Тогава

$$\begin{aligned}
R(n) &= \sqrt{n}R(n-1) + 1 \\
&= \sqrt{n}(\sqrt{n-1}R(n-2) + 1) + 1 \\
&= \sqrt{n(n-1)}R(n-2) + \sqrt{n} + 1 \\
&= \sqrt{n(n-1)}(\sqrt{n-2}R(n-3) + 1) + \sqrt{n} + 1 \\
&= \sqrt{n(n-1)(n-2)}R(n-3) + \sqrt{n(n-1)} + \sqrt{n} + 1 \\
&\cdots \\
&= \sqrt{n(n-1)\cdots 2}R(1) + \sqrt{n(n-1)\cdots 3} + \cdots + \sqrt{n(n-1)(n-2)} + \sqrt{n(n-1)} + \sqrt{n} + 1
\end{aligned}$$

Тривиално се показва, че $R(n) \asymp \sqrt{n(n-1)(n-2)\cdots 2}$: изваждаме този множител пред скоби в последния израз и забелязваме, че сумата от дробите, която се получава в скобите, е ограничена от константа. Тогава $R(n) \asymp \sqrt{n!}$. □

Зад. 3 Разгледайте алгоритъма AlgX.

```
AlgX(A[1,...,n]: масив от цели числа, две по две различни; n ≥ 1)
 1  X ← True
 2  Y ← True
 3  for i ← 1 to n − 1
 4      if X
 5          if A[i] > A[i + 1]
 6              X ← False
 7      if not X
 8          if A[i] < A[i + 1]
 9              Y ← False
10  return Y
```

- Обяснете на прост български какво прави AlgX.
- Докажете това, използвайки инварианта на цикъла.

Забележка Ред 7 на алгоритъма да се чете като "**else**". Това е **else** на **if**-а на ред 4. За работата на алгоритъма няма значение дали е по този начин, или по начина в кода горе, но се чете и възприема по-лесно с **else**.

Решение AlgX връща True тогава и само тогава, когато съществува индекс i, такъв че $1 \leq i \leq n$ и подмасивът $A[1,\ldots,i]$ е растящ, а подмасивът $A[i,\ldots,n]$ е намаляващ. Такъв масив се нарича унимодален. В някакъв смисъл, "унимодален" е обобщение на "растящ" и "намаляващ" – те са негови частни случаи. Понятието "унимодален" не е изучавано в час и не се иска отговорът да го съдържа – това е само за Ваше сведение. За удобство тук приемаме, че едноелементен масив е както растящ, така и намаляващ, така че едноелементен масив е унимодален. Дали празният масив е унимодален или не, няма да спекулираме – по условие $n \geq 1$ и няма да разглеждаме празни подмасиви на $A[]$.
По-формално, масивът е унимодален тогава и само тогава, когато

$$\exists i \in \{1,\ldots,n\} : A[1] < A[2] < \cdots < A[i-1] < A[i] > A[i+1] > \cdots > A[n-1] > A[n]$$

И така, AlgX връща True тогава и само тогава, когато входът е унимодален масив. Сега ще докажем това с инварианта.

Следната инварианта е в сила. При всяко достигане на ред 3:

- X е True тогава и само тогава, когато $A[1,\ldots,i]$ е растящ.
- Y е True тогава и само тогава, когато $A[1,\ldots,i]$ е унимодален.

При $i = 1$ и двете са в сила: от една страна, $X =$ True и $Y =$ True заради присвояванията на редове 1 и 2. От друга страна, $A[1,\ldots,i]$ е $A[1]$, който по дефиниция е унимодален.✓

Да разгледаме произволно достигане на ред 3, което не е последното, и да допуснем, че твърденията са верни. Изпълнението отива на ред 4. Ще разгледаме три случая, които се изключват взаимно и са изчерпателни.

Случай 1. $A[1,\ldots,i]$ е растящ. Тогава X е True и Y е True от допускането. Щом X е True, изпълнението минава на ред 5.

Ако $A[i] < A[i+1]$, то от една страна подмасивът $A[1,\ldots,i+1]$ е растящ, а от друга страна булевото условие на ред 5 е лъжа и изпълнението се връща на ред 3, като стойностите на X и Y се запазват, а индексната променлива i се инкрементира. Спрямо новата стойност на i можем да твърдим, че $A[1,\ldots,i]$ е растящ. Виждаме, че в този подслучай инвариантата остава вярна.

Ако $A[i] > A[i+1]$, то от една страна подмасивът $A[1,\ldots,i+1]$ не е растящ, но е унимодален, а от друга страна булевото условие на ред 5 е истина. Тогава на X се присвоява False на ред 6. После изпълнението се връща на ред 3, като стойността на Y се запазва, а индексната променлива i се инкрементира. Спрямо новата стойност на i можем да твърдим, че $A[1,\ldots,i]$ е унимодален, без да е растящ. Виждаме, че в този подслучай инвариантата остава вярна.

Случай 2. $A[1,\ldots,i]$ не е растящ, но е унимодален. Тогава X е False и Y е True от допускането. Щом X е False, изпълнението минава на ред 7.

Ако $A[i] < A[i+1]$, то от една страна подмасивът $A[1,\ldots,i+1]$ не е унимодален, а от друга страна булевото условие на ред 8 е истина. Тогава на Y се присвоява False на ред 9. После изпълнението се връща на ред 3, като и X, и Y са False, а индексната променлива i се инкрементира. Спрямо новата стойност на i можем да твърдим, че $A[1,\ldots,i]$ ke е унимодален. Виждаме, че в този подслучай инвариантата остава вярна.

Ако $A[i] > A[i+1]$, то от една страна подмасивът $A[1,\ldots,i+1]$ е унимодален, а от друга страна булевото условие на ред 8 е лъжа. Изпълнението се връща на ред 3, като стойностите на X и Y се запазват, а индексната променлива i се инкрементира. Спрямо новата стойност на i можем да твърдим, че $A[1,\ldots,i]$ е унимодален. Виждаме, че в този подслучай инвариантата остава вярна.

Случай 3. $A[1,\ldots,i]$ не е унимодален (което означава, че не е растящ). Тогава X е False и Y е False от допускането. Щом X е False, изпълнението минава на ред 7.

От една страна, независимо от изхода на сравнението на ред 7, X и Y запазват стойностите си, защото Y вече е False. От друга страна, независимо от изхода на сравнението на ред 7, масивът $A[1,\ldots,i+1]$ не е унимодален, понеже $A[1,\ldots,i]$ не е унимодален. Изпълнението се връща на ред 3, като стойностите на X и Y се запазват, а индексната променлива i се инкрементира. Спрямо новата стойност на i можем да твърдим, че $A[1,\ldots,i]$ не е унимодален. Виждаме, че в този подслучай инвариантата остава вярна.

При терминацията на алгоритъма $i = n$. Заместваме i с n във втората клауза на инвариантата и получаваме:

- Y е True тогава и само тогава, когато $A[1,\ldots,n]$ е унимодален.

Веднага следва, че стойността, която алгоритъмът връща на ред 10 показва дали входът е унимодален или не □

Зад. 4 Даден е масив $A[1,2,\ldots,n]$. Всеки елемент на масива има атрибут цвят (може да има и други атрибути), който е или бял, или сив, или черен. Има поне по един елемент от всеки цвят. Допустимите функции за достъп до масива са:

- Colour(i), където $1 \le i \le n$. Връща цвета на $A[i]$, без да променя нищо в масива.
- Swap(i,j), където $1 \le i \le n$, $1 \le j \le n$ и $i \ne j$. Разменя $A[i]$ и $A[j]$ в масива, без да променя останалите елементи. Не връща нищо.

Други видове достъп до масива не са позволени. Предложете алгоритъм с колкото е възможно по-ниска (в асимптотичния смисъл) сложност по време и по памет, който пренарежда масива по такъв начин, че да започва (отляво надясно) с белите елементи, после да са сивите, и най-накрая да са черните.

Решение: Използваме функцията Partition на Quick Sort, било във варианта Hoare, било във варианта Lomuto. Важното е да работи в линейно време и да е in-place. Една малка модификация: за pivot избираме кой да е сив елемент – по условие, поне един такъв има. В линейно време и с константна допълнителна памет можем да намерим сив елемент. Ползвайки него за pivot, викаме Partition, като ключовете са цветовете, върху които е дефинирана наредбата бял преди сив преди черен. Очевидно двете функции Colour(i) и Swap(i,j) са напълно достатъчни, за да реализираме Partition. След приключването на работата на Partition със сигурност белите ще са преди сивите и сивите ще са преди черните.

Сложността по време е $\Theta(n)$, а сложността по памет е $\Theta(1)$. И двете сложности са асимптотично оптимални. □

Зад. 5 Наредете по асимптотично нарастване следните шест функции. Обосновете отговорите си кратко. Напишете в явен вид самата наредба.

$f_1 = (\lg n)^{\lg n}$, $f_2 = \left(\lg\left(\sqrt{n}\right)\right)^{\lg\left(\sqrt{n}\right)}$, $f_3 = (\lg n)^{(\lg n)^{\lg n}}$

$f_4 = 2^{2^{n-1}}$, $f_5 = \sum_{k=0}^{n} \binom{n}{k}$, $f_6 = \binom{2n}{n}$

Решение:

(i) Ще докажем, че $f_2 \prec f_1$. Не е добра идея да се логаритмуват двете функции, защото образите след логаритмичната трансформация имат една и съща асимптотика и това не ни казва нищо за отношението между f_2 и f_1.

Забелязваме, че

$$f_2 = \left(\lg\left(\sqrt{n}\right)\right)^{\frac{1}{2}\lg n} = \sqrt{\left(\lg\left(\sqrt{n}\right)\right)^{\lg n}}$$

Но очевидно

$$\left(\lg\left(\sqrt{n}\right)\right)^{\lg n} \preceq (\lg n)^{\lg n}$$

а от друга страна

$$\sqrt{\left(\lg\left(\sqrt{n}\right)\right)^{\lg n}} \prec \left(\lg\left(\sqrt{n}\right)\right)^{\lg n}$$

понеже става дума за растяща функция и корен-втори от нея. Тогава

$$\sqrt{\left(\lg\left(\sqrt{n}\right)\right)^{\lg n}} \prec (\lg n)^{\lg n}$$

или, с други думи, $f_2 \prec f_1$.

(ii) Ще докажем, че $f_1 \prec f_5$. Забелязваме, че $f_5 = 2^n$ заради Теоремата на Нютон. Сравняваме $(\lg n)^{\lg n}$ с 2^n. Логаритмуваме двете функции. Лявата става $(\lg n)(\lg\lg n)$. Дясната става n. Както знаем, $(\lg n)(\lg\lg n) \prec n$. Желаният резултат следва веднага.

(iii) Ще докажем, че $f_5 \prec f_6$. Лесно се доказва, че $f_6 \asymp \frac{4^n}{\sqrt{n}}$ – доказателство, ползващо апроксимацията на Stirling, има в сборника със задачи. Тривиално се доказва, че $\lim_{n\to\infty} \frac{\frac{4^n}{\sqrt{n}}}{2^n} = \infty$. Желаният резултат следва веднага.

(iv) Ще докажем, че $f_6 \prec f_3$. Както вече видяхме, $f_6 \asymp \frac{4^n}{\sqrt{n}}$. Сравняваме $\frac{4^n}{\sqrt{n}}$ и f_3, логаритмувайки и двете. Лявата функция след логаритмуването има асимптотика $\Theta(n)$. Дясната има асимптотика $\left((\lg n)^{\lg n}\right)(\lg n)$. Очевидно, $\left((\lg n)^{\lg n}\right)(\lg n)$ расте по-бързо, в асимптотичния смисъл, от всяка полиномиална функция. Желаният резултат следва веднага.

(v) Ще докажем, че $f_3 \prec f_4$. Логаритмуваме двете функции и вече сравняваме $\left((\lg n)^{\lg n}\right)(\lg n)$ с 2^{n-1}. Дясната от тези расте по-бързо в асимптотичния смисъл – факт, който може да установим с още едно логаритмуване. Желаният резултат следва веднага.

От **(i)**–**(v)** следва наредбата:

$$f_2 \prec f_1 \prec f_5 \prec f_6 \prec f_3 \prec f_4$$

□

Зад. 6 Да си представим двама души, единият от които, да кажем X, знае някакво мултимножество от цели числа $\{a_1, a_2, \ldots, a_n\}_M$, а другият, да кажем Y, иска да установи някаква сортирана последователност от тези числа, задавайки въпроси на X. Y няма директен достъп до числата, а само може да задава въпроси за тях и да получава отговори. Може да има повтарящи се стойности, тоест може $a_i = a_j$ за различни i и j, така че сортираната последователност може да не е уникална.

Y има право да задава на X въпроси от вида $\boxed{\textit{Дали } a_i \leq a_j \textit{?}}$, където i и j са валидни индекси, и никакъв друг вид въпроси. Очевидно отговорите са еднобитови – да или не.

Правилата са такива: първо Y задава въпросите си, после X отговаря на всички въпроси, и повече въпроси не се задават – Y вече трябва да знае със сигурност някаква сортирана последователност на числата.

Задачата е:

- Докажете, че $\Omega(n^2)$ е асимптотична долна граница за броя на въпросите, които Y задава.
- Как мислите, тази долна граница не е ли в противоречие с факта, че съществуват сортиращи алгоритми, базирани на директни сравнения и работещи във време $O(n \lg n)$?

Решение Въпросите трябва да са поне $\binom{n}{2}$. А именно, за всяко двуелементно подмножество $\{a_i, a_j\}$, трябва да бъде зададен поне един въпрос с a_i и a_j. Да допуснем противното: не задаваме въпрос за някои a_i и a_j, където $i \neq j$. Без ограничение на общността, нека $a_i < a_j$. Ако тези числа са съседни по стойност, тоест няма a_k, такова че $a_i < a_k < a_j$, питащият Y няма как да разбере дали $a_i < a_j$ или не, ако не ги сравни директно (с въпрос за точно тях). Защото, ако можеше да се намери сортирана последователност без да задаваме въпрос за тях, при размяна на стойностите им ще бъде дедуцирана същата сортирана последователност, но тя вече не е правилна, защото $a_i > a_j$ след размяната на стойностите.

А противоречие с горната граница $O(n \lg n)$ за задачата Сортиране няма. Наистина, сортиращите алгоритми, базирани на директни сравнения, "дедуцират" сортираната последователност с $O(n \lg n)$ въпроса, но схемата, по която те задават въпросите си, е *адаптивна*. С други думи, това кои елементи биват сравнявани зависи от отговорите на предни сравнения. В тази задача схемата от въпроси *не е* адаптивна – по условие, първо Y задава всички въпроси и чак след това получава отговори, като не може да задава повече въпроси.

□