Решения на контролното по ДАА от 13.04.2013 г.

Зад. 1 Нека $f(n)$ и $g(n)$ произволни асимптотично положителни функции с домейн $\mathbb{N}$. Докажете или опровергайте всяко от следните твърдения:

а) Поне едно от $f(n) = O(g(n))$ и $g(n) = O(f(n))$ е в сила.

б) Ако $f(n) = O(n)$, то $g(f(n)) = O(g(n))$.

Решение, а): Твърдението не е вярно. Контрапример е $f(n) = n$ и

$$g(n) = \begin{cases} n^2, & \text{ако } n \text{ е четно} \\ 1, & \text{в противен случай} \end{cases}$$

Решение, б): Твърдението не е вярно. Контрапример е $f(n) = 2n$ и $g(n) = 2^n$. □

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

$$10 \lg n, \quad 20 \lg n!, \quad \frac{2^n}{\sqrt{n} + 3\sqrt[3]{n + \lg n}}, \quad (1.99)^n, \quad n^2$$

$$\sum_{k=1}^{n} \frac{1}{k}, \quad \binom{n^2}{\frac{1}{2}n^2}, \quad (n^2)!, \quad (n!)^2, \quad n^{\frac{1}{\lg \lg n}}$$

Решение:

(i) Ще докажем, че $(n^2)! \succ \binom{n^2}{\frac{1}{2}n^2}$. Съгласно апроксимацията на Стирлинг,

$$(n^2)! \approx \sqrt{2\pi n^2} \frac{(n^2)^{n^2}}{e^{n^2}} \asymp n^{2n^2+1} e^{-n^2} \tag{1}$$

Ще докажем, че

$$\binom{m}{\frac{m}{2}} \asymp \frac{1}{\sqrt{m}} 2^m \tag{2}$$

Действително, използвайки апроксимацията на Стирлинг и приемайки, че m е четно, имаме

$$\binom{m}{\frac{m}{2}} = \frac{m!}{\left(\frac{m}{2}\right)!\,\left(\frac{m}{2}\right)!} \approx \frac{\sqrt{2\pi m}\frac{m^m}{e^m}}{\left(\sqrt{2\pi\frac{m}{2}}\frac{\left(\frac{m}{2}\right)^{\frac{m}{2}}}{e^{\frac{m}{2}}}\right)^2} \asymp \frac{m^{m+\frac{1}{2}}\, e^{-m}}{\left(\sqrt{m}\sqrt{m}^m\, \sqrt{2}^{-m}\, \sqrt{e}^{-m}\right)^2}$$

$$= \frac{m^{m+\frac{1}{2}}\, e^{-m}}{m m^m\, 2^{-m}\, e^{-m}} = \frac{1}{\sqrt{m}} 2^m$$

Заместваме m с n^2 в (2) и получаваме

$$\binom{n^2}{\frac{1}{2}n^2} \asymp \frac{1}{n} 2^{n^2} \tag{3}$$

За да докажем твърдението, достатъчно е да приложим (1) и (3) и да изследваме границата:

$$\lim_{n\to\infty} \frac{n^{2n^2+1} e^{-n^2}}{\frac{1}{n} 2^{n^2}} = \lim_{n\to\infty} n^2 \left(\frac{n^2}{2e}\right)^{n^2} = \infty$$

(ii) Ще докажем, че $\binom{n^2}{\frac{1}{2}n^2} \succ (n!)^2$. Съгласно апроксимацията на Стирлинг,

$$(n!)^2 \approx \left(\sqrt{2\pi n}\frac{n^n}{e^n}\right)^2 \asymp n^{2n+1}e^{-2n} \tag{4}$$

Имайки предвид (3) и (4), достатъчно е да разгледаме логаритмите

$$\lg\left(\frac{1}{n}2^{n^2}\right) \asymp n^2 \tag{5}$$

$$\lg\left(n^{2n+1}e^{-2n}\right) \asymp n\lg n \tag{6}$$

Тъй като $n^2 \succ n\lg n$, от (5) и 6 заключаваме, че $\binom{n^2}{\frac{1}{2}n^2} \succ (n!)^2$.

(iii) Ще докажем, че $(n!)^2 \succ \frac{2^n}{\sqrt{n}+3\sqrt[3]{n+\lg n}}$, доказвайки, че $(n!)^2 \succ 2^n$ и $2^n \succ \frac{2^n}{\sqrt{n}+3\sqrt[3]{n+\lg n}}$. За първото от тези твърдения ще използваме вече доказания факт (4). Действително,

$$\lim_{n\to\infty}\frac{n^{2n+1}e^{-2n}}{2^n} = \lim_{n\to\infty} n\, n^{n+1}\,\frac{n^n}{e^{2n}\,2^n} = \lim_{n\to\infty} n\, n^{n+1}\,\left(\frac{n}{e^2\,2}\right)^n = \infty$$

Второто твърдение също можем да докажем с граници:

$$\lim_{n\to\infty}\frac{2^n}{\frac{2^n}{\sqrt{n}+3\sqrt[3]{n+\lg n}}} = \lim_{n\to\infty}\sqrt{n}+3\sqrt[3]{n+\lg n} = \infty$$

(iv) Ще докажем, че $\frac{2^n}{\sqrt{n}+3\sqrt[3]{n+\lg n}} \succ (1.99)^n$. Първо забелязваме, че $\sqrt{n}+3\sqrt[3]{n+\lg n} \asymp \sqrt{n}$, понеже

$$\lim_{n\to\infty}\frac{\sqrt{n}}{\sqrt{n}+3\sqrt[3]{n+\lg n}} = \lim_{n\to\infty}\frac{1}{1+\frac{3\sqrt[3]{n+\lg n}}{\sqrt{n}}} = \lim_{n\to\infty}\frac{1}{1+0} = 1$$

За да докажем желаното твърдение, достатъчно е да изследваме границата

$$\lim_{n\to\infty}\frac{\frac{2^n}{\sqrt{n}+3\sqrt[3]{n+\lg n}}}{(1.99)^n} = \lim_{n\to\infty}\frac{\frac{2^n}{(1.99)^n}}{\sqrt{n}+3\sqrt[3]{n+\lg n}} = \lim_{n\to\infty}\frac{\left(\frac{2}{1.99}\right)^n}{\sqrt{n}+3\sqrt[3]{n+\lg n}} = \infty$$

Границата е безкрайност, понеже в числителя на последния израз има експоненциална функция с основа по-голяма от 1, а знаменателят е асимптотично еквивалентен на полиномиална функция.

(v) Това, че $(1.99)^n \succ n^2$, следва тривиално от факта, че всяка експоненциална функция с основа по-голяма от 1 расте асимптотично по-бързо от всяка полиномиална функция.

(vi) Ще докажем, че $n^2 \succ 20\lg n!$. Известно е, че

$$\lg n! \asymp n\lg n \tag{7}$$

Желаният резултат следва веднага, ако изследваме границата

$$\lim_{n\to\infty}\frac{n^2}{n\lg n} = \lim_{n\to\infty}\frac{n}{\lg n} = \infty$$

припомняйки си, че всяка полиномиална функция расте по-бързо от всяка полилогаритмчна функция.

(vii) Ще докажем, че $20\lg n! \succ n^{\frac{1}{\lg\lg n}}$. Имайки предвид (7), достатъчно е да докажем, че $n\lg n \succ n^{\frac{1}{\lg\lg n}}$. Действително, ако логаритмуваме двете функции, получаваме

$$\lg(n\lg n) \asymp \lg n$$

$$\lg\left(n^{\frac{1}{\lg\lg n}}\right) \asymp \frac{\lg n}{\lg\lg n}$$

От факта, че $\lg n \succ \frac{\lg n}{\lg \lg n}$, желаният резултат следва веднага.

(viii) Фактът, че $\sum_{k=1}^{n} \frac{1}{k} \asymp \lg n$, е доказан на упражнения и лекции, следователно $10 \lg n \asymp \sum_{k=1}^{n} \frac{1}{k}$.

(ix) Ще докажем, че $n^{\frac{1}{\lg \lg n}} \succ 10 \lg n$. Действително, ако логаритмуваме двете функции, получаваме съответно $\frac{\lg n}{\lg \lg n}$ и $\lg \lg n + \lg 10$. Фактът, че $\frac{\lg n}{\lg \lg n} \succ \lg \lg n$ следва почти директно от факта, че всяка полиномиална функция расте по-бързо от всяка логаритмична.

От **(i)**–**(ix)** следва наредбата:

$$(n^2)! \succ \binom{n^2}{\frac{1}{2}n^2} \succ (n!)^2 \frac{2^n}{\sqrt{n} + 3\sqrt[3]{n} + \lg n} \succ (1.99)^n \succ n^2 \succ 20 \lg n! \succ n^{\frac{1}{\lg \lg n}} \succ \sum_{k=1}^{n} \frac{1}{k} \asymp \lg n$$

□

Зад. 3 Следните два програмни фрагмента са написани на `C`. И в двата случая $A[0, 1, \ldots, n-1]$ е масив от цели числа и $n \geq 2$. Разгледайте функцията `func1( )` и докажете, че тя връща средното аритметично от елементите на масива. Разгледайте функцията `func2( )`. Първо определете какво прави тя и после докажете това свойство чрез инварианта на цикъла. Във `func2( )` има викания към две функции. Функкцията `sort( )` е произволна сортираща функция с първи аргумент указател към масив и втори аргумент, броят на елементите на масива, а `abs( )` с аргумент-цяло число връща абсолютната стойност на числото. Коректността на `sort( )` и на `abs( )` може да считате за даденост.

а)

```
int A[n];
float func1(int n) {
   int i, s = 0;
   for(i = 0; i < n; i ++) {
        s += A[i]; }
   return (float)s/n; }
```

б)

```
int A[n];
int func2(int n) {
   int i, m;
   sort(A, n);
   m = abs(A[0] - A[1]);
   for(i = 1; i < n-1; i ++) {
      if (abs(A[i] - A[i+1]) < m) {
         m = abs(A[i] - A[i+1]); } }
   return m; }
```

Решение, а) Инварианта за `func1( )` е

При всяко достигане на ред 4, $s = \sum_{j=0}^{i-1} A[j]$.

База При първото достигане на ред 4, $s = 0$ заради присвояването на ред 3. От друга страна, $\sum_{j=0}^{i-1} A[j] = 0$, защото $i = 0$, което на свой ред означава, че множеството $\{0, \ldots, i-1\}$ е празно, а сума, в която индексната променлива взема стойности от празното множество, е нула. ✓

Запазване Нека твърдението е вярно при някое достигане, което не е последното. Преди присвояването на ред 5, имаме $s = \sum_{j=0}^{i-1} A[j]$ от предположението. След присвояването на ред 5, имаме $s = \left(s = \sum_{j=0}^{i-1} A[j]\right) + A[i]$. При следващото достигане на ред 4, i се инкрементира с единица, което означава, че по отношение на новото i имаме $s = \sum_{j=0}^{i-1} A[j]$.

Терминация При последното достигане на ред 4, в сила е

$$i = n$$
$$s = \sum_{j=0}^{i-1} A[j]$$

Следователно, $s = \sum_{j=0}^{n-1} A[j]$ в края на изпълнението на цикъла. Тогава на ред 5, $s = \frac{\sum_{j=0}^{n-1} A[j]}{n}$, което е именно средно аритметичното от елементите на масива.

Решение, б) Тъй като не е казано, че елементите на масива имат различни големини, може да има елементи с еднакви големини. Казвайки "различни елементи" ще имаме предвид елементи на различни позиции в масива, а не елементи, които имат непременно различни големини.

Функцията `func2( )` връща $\min\{|A[j] - A[k]| : 0 \leq j < k \leq n-1\}$, тоест минималната разлика между големини на различни елементи от масива. Първо ще докажем, че такава минимална разлика се реализира между двойка (различни) елементи, които са съседи в някоя сортирана последователност. Ако допуснем, че минималната разлика между големини се реализира между два елемента x и y, такива че във всяка сортирана последователност между тях има елемент поне един елемент z, очевидно големината на z е строго между x и y, така че между x и z се реализира дори по-малка разлика от големини, отколкото между x и y. Инварианта за `func2( )` е

При всяко достигане на ред 6, m съдържа $\min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$.

Естествено, инвариантата е формулирана спрямо сортираната последователност, а не спрямо оригиналния масив $A[]$. Тъй като допуснахме, че викането на `sort(A, n)` на ред 4 сортира масива, в доказателството надолу ще считаме, че масивът е сортиран.

База При първото достигане на ред 6, $m = \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$ понеже $i = 1$ заради присвояването на ред 6 и $m = |A[0] - A[1]|$ заради присвояването на ред 5. ✓

Запазване Нека твърдението е вярно при някое достигане, което не е последното. В сила е

$$\min\{|A[j] - A[k]| : 0 \leq j < k \leq i+1\} \neq \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\} \leftrightarrow$$
$$\min\{|A[j] - A[k]| : 0 \leq j < k \leq i+1\} < \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\} \leftrightarrow$$
$$|A[i] - A[i+1]| < \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$$

Съгласно индукционното предположение, в началото на изпълнението на цикъла m съдържа именно $\min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$.

Да допуснем, че $\min\{|A[j] - A[k]| : 0 \leq j < k \leq i+1\} = \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$. Тогава условието на ред 7 не е истина и m не променя стойността си. Изразено в термините на старото i, $m = \min\{|A[j] - A[k]| : 0 \leq j < k \leq i+1\}$. След инкрементирането на i, $m = \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$. Следователно, в този случай инвариантата е в сила при следващото достигане на ред 6.

Да допуснем, че $\min\{|A[j] - A[k]| : 0 \leq j < k \leq i+1\} \neq \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$. Тогава, както казахме, $|A[i] - A[i+1]| < \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$, което е същото като $|A[i] - A[i+1]| < m$ съгласно индукционното предположение. Но в такъв случай условието на ред 7 е истина и m променя стойността си, ставайки $|A[i] - A[i+1]|$. Изразено в термините на старото i, $m = \min\{|A[j] - A[k]| : 0 \leq j < k \leq i+1\}$. След инкрементирането на i, $m = \min\{|A[j] - A[k]| : 0 \leq j < k \leq i\}$. Следователно, и в този случай инвариантата е в сила при следващото достигане на ред 6.

Терминация При последното достигане на ред 6, $i = n - 1$, следователно

$$m = \min\{|A[j] - A[k]| : 0 \leq j < k \leq n-1\}$$

□

Зад. 4 Масивът $A[1, \ldots, n]$ е макс-пирамида и $n \geq 7$. Елементите на $A[]$ са два по два различни различни. Докажете, че трите най-големи елемента на масива се намират в подмасива $A[1 \ldots, 7]$.

Решение: Да допуснем противното, а именно, че елемент измежду трите най-големи е на позиция $i \geq 8$. Да си припомним, че родителят на елемент j е на позиция $p(j) = \left\lfloor \frac{j}{2} \right\rfloor$. Тогава родителят на елемент i е на позиция $p(i)$, където $p(i) \geq 4$. Тъй като $A[]$ е макс-пирамида от различни елементи, родителят на третия по големина елемент е един от двата най-големи елемента. Току-що изведохме, че той е на позиция $k \geq 4$. Тогава родителят на елемент k е на позиция $t \geq 2$. Но родителят на втория по големина елемент в макс-пирамида от различни елементи може да е само най-големият елемнт. Изведохме, че най-големият елемент в макс-пирамида от различни елементи е на позиция $t \geq 2$. Но това е невъзможно – знаем, че в такава пирамида максималният елемент е точно на първа позиция. Следователно, допускането ни е невярно. □

Зад. 5 Докажете, че алгоритимът Split намира k най-малки елемента на масива $A[1, \ldots, n]$ и ги слага в подмасива $A[1, \ldots, k]$, а останалите елементи слага в подмасива $A[k+1, \ldots, n]$.

```
Split(A[1, 2, ..., n]: array; k: index in A)
1  l ← 1
2  h ← n
3  m ← n + 1
4  while k ≠ m do
5      m ← Partition(A, l, h)
6      if m < k
7          l ← m + 1
8      else
9          h ← m − 1
```

Решение: Да разгледаме следната инвариантна формула.

> При всяко достигане на ред 4 е изпълнено точно едно от следните две твърдения (подусловия):
>
> **(1)** Всички елементи на подмасива $A[1 \dots l-1]$ са по-малки или равни на елементите на $A[l \dots h]$, които пък са по-малки или равни на елементите на подмасива $A[h+1 \dots n]$. Освен това стойността на параметъра k е между l и h ($l \leq k \leq h$), $l \leq h$ и $k \neq m$.
>
> **2)** Изпълнено е равенството $k = m$ и всички елементи на подмасива $A[1 \dots k{-}1]$ са по-малки или равни на $A[k]$, което пък е по-малко или равно на всички елементи на подмасива $A[k+1 \dots n]$.

База При първото влизане в ред 4, подмасивите $A[1, \dots, l{-}1]$ и $A[h{+}1, \dots, n]$ са празни, а $A[l, \dots, h]$ е целият масив ($l = 1,\ h = n$), k е между l и h, тъй като е индекс във входния масив, следователно е вярно подусловие **(1)** на инвариантата. Освен това $k < m$, следователно подусловие **(2)** на инвариантата не е вярно.

Запазване Да допуснем, че инвариантната формула е вярна при някое достигане на ред 4, което не е последното. Тогава $k \neq m$, следователно измежду двете подусловия е изпълнено точно **(1)**, следователно $l \leq k \leq h$. Тялото на цикъла ще се изпълни, като най-напред ще бъде присвоена нова стойност на $m = \textsc{Partition}(A, l, h)$. $\textsc{Partition}$ работи така, че $A[m]$ ще раздели подмасива $A[l, \dots, h]$ на малки и големи елементи спрямо разделителя $A[m]$, като малките ще се разположат наляво, а големите – надясно от $A[m]$. За k има три възможности:

(i) $k < m$. От $l \leq k$ следва, че $l < m$. Елементите на подмасива $A[l, \dots, m-1]$ са по-малки от елементите на $A[h+1, \dots, n]$, защото е вярно подусловие **(1)**, а освен това са по-малки и от елементите на подмасива $A[m, \dots, h]$ (поради изпълнението на $\textsc{Partition}$), следователно елементите на $A[l, \dots, m-1]$ са по-малки или равни на елементите на подинтервала $A[m, \dots, n]$. Ще се изпълни ред 8. За новата стойност на $h = m - 1$ се изпълняват точно всички изисквания на подусловие **(1)**. Следователно, при новото достигане на ред 4 в сила ще бъде точно подусловие **(1)**.

(ii) $k > m$. Този случай е симетричен на горния. Ще се изпълни ред 7, но пак ще се запази верността на подусловие **(1)**, така че при новото достигане на ред 4 в сила ще бъде точно подусловие **(1)**.

(iii) $k = m$. В този случай ще стане вярно твърдение **(2)**, а **(1)** ще престане да бъде вярно. Наляво от $A[k] = A[m]$ имаме два подинтервала: $A[1, \dots, l-1]$ и $A[l, \dots, k-1]$. Елементите на първия са по-малки или равни на $A[k]$ поради подусловие **(1)** на индукционното предположение, а елементите на $A[l, \dots, k-1]$ са по-малки или равни на $A[m]$ поради изпълнението на $\textsc{Partition}$. Следователно елементите на обединението им $A[1, \dots, k-1]$ са по-малки или равни на $A[k]$.

С аналогични разсъждения установяваме, че $A[k]$ е по-малко или равно на елементите на подинтервала $A[k+1 \dots n]$.

Терминация При последното достигане на ред 4, $k = m$. В този случай е изпълнено подусловие **(2)** и задачата е решена, тъй като всички елементи наляво от $A[k]$ са по-малки или равни на $A[k]$, а всички надясно от $A[k]$ са по-големи или равни нему, тоест подмасивът $A[1 \dots k]$ съдържа най-малките k елемента на оригиналния масив.

С направените дотук разсъждения установихме, че инвариантната формула е изпълнена при всяко влизане в ред 4. Остава да проверим дали алгоритъмът спира, тоест дали **while**-цикълът се изпълнява краен брой пъти. Лесно се вижда, че цикълът се изпълнява докато е вярно подусловие **(1)** на инвариантната формула. При всяко преминаване през тялото на цикъла дължината на интервала $[l, \dots, h]$ намалява поне с единица. Следователно след най-много $n-1$ изпълнения на цикъла интервалът ще стане с дължина 1 и тогава ще се достигне равенство $m = k$ (ако това не стане по-рано). При първото достигане на равенството $m = k$ програмата ще прекрати изпълнението си и задачата ще бъде решена поради верността на подусловие **(2)**. □

Зад. 6 Нека $a_1\ a_2\ \cdots\ a_n$ е пермутация на множеството $\{1, 2, \dots, n\}$. *Инверсия* в тази пермутация се нарича всяка наредена двойка (i, j), такава че $1 \leq i < j \leq n$ и $a_i > a_j$. *Инверсният вектор на пермутацията* $a_1\ a_2\ \cdots\ a_n$ е векторът $(b_1, b_2, \dots, b_n)$, където $\forall i, 1 \leq i \leq n$, b_i е броят на елементите в $a_1\ a_2 \cdots\ a_n$, които са вляво от i и са по-големи от i.

а) Напишете инверсния вектор на пермутацията 4 2 3 7 1 8 5 9 6.

б) Предложете алгоритъм, който по зададен инверсен вектор да извежда оригиналната пермутация. Допуснете, че входът $(b_1, b_2, \dots, b_n)$ на алгоритъма е коректен инверсен вектор на някоя пермутация на числата $\{1, 2, \dots, n\}$. Дайте кратка обосновка на коректността на Вашия алгоритъм (не се иска строго доказателство по индукция или с инварианта) и изследвайте сложността му по време.

Решение, а) $(4, 1, 1, 0, 2, 3, 0, 0, 0)$.

Решение, б) Да разсъждаваме така. Елемент с големина k от пермутацията, която искаме да построим, може да има между 0 и $n-k$ числа вляво от себе си и по-големи от себе си в тази пермутация.

Знаем, че числото n е елемент на пермутацията (която искаме да построим). За b_n има само една възможност – да бъде нула. Записваме n в списък. За b_{n-1} има две възможности. Ако b_{n-1} е нула, слагаме $n-1$ вляво от n в списъка. В противен случай, вдясно. За b_{n-2} има три възможности. Ако b_{n-2} е нула, слагаме $n-2$ вляво от вече сложените два елемента в списъка. Тоест, на позиция 0. Ако b_{n-2} е единица, слагаме $n-2$ между тях. Ако е двойка, слагаме $n-2$ вдясно от тях. И така нататък. Използвайки тази идея, елемент k ще бъде сложен в списъка така, че точно b_k елемента да са вляво от него. Накрая ще разгледаме b_1 и ще сложим елемент с големина 1 на такава позиция в списъка, че b_1 елемента са вляво от него. След което списъкът ще представлява търсената пермутация. Следният алгоритъм имплементира тази идея.

GenerativePermutation$((b_1, b_2, \dots, b_n)$: inversion vector$)$

нека X е списък от числа
for $k \leftarrow n$ **downto** 1
    сложи k в X така, че b_k елемента да са вляво от него
return X

Коректността на алгоритъма следва от горните разсъждения. Сложността му по време е квадратична, тъй като външният **for**-цикъл се изпълнява точно n пъти, а вътрешният (имплицитен) цикъл се изпълнява в най-лошия случай k пъти, така че в най-лошия случай сложността е пропорционална на

$$\sum_{k=1}^{n} \sum_{j=1}^{k} 1 = \sum_{k=1}^{n} k = \Theta(n^2)$$

□