Семестриално контролно по ДАА (избираема), СУ, ФМИ, 26. 11. 2015 г.

Име: _____, ФН:_____, Спец.:____, Курс:___

Задача	1	2	3	4	5	6	Общо
получени точки							
максимум точки	20	20	20	20	20	30	130

Забележка: Всеки път, когато сортирате данни, уточнявайте коя сортировка използвате!

Задача 1. Дадени са времевите сложности на няколко алгоритъма:
$9\,.\,5^{\,n}$, $n^{\,7}\,\log(8n+14)$, $7\,.\,5^{\,n^2}$, $(5n)^{\,7n}$, $\sqrt{n^{\,7}}$.
Подредете алгоритмите по бързодействие (най-бързия на първо място). Отговорът да се обоснове!

Задача 2. Да се решат рекурентните уравнения:

а) $T(n) = 10\,T(n-1) - 16\,T(n-2) + 8^{\,n}$;

б) $T(n) = T(n-1) + \sqrt{n^{\,81}}$;

в) $T(n) = 6^{\,n} + \sum\limits_{k\,=\,0}^{n-1} T(k)$;

г) $T(n) = 243\,T\left(\dfrac{n}{3}\right) + 9n^4$.

Задача 3. Имаме следния алгоритъм:

MyFunc(n: integer)
```
1  s ← 4
2  while s ≤ n do
3      s ← 8 × s + 6
4  return s
```
а) Каква стойност връща алгоритъмът?
б) Оценете по порядък времевата сложност на алгоритъма.

Задача 4. Даден е масив от n цели положителни числа, които могат да бъдат много големи. Съставете възможно най-бърз алгоритъм, който да намира най-малкото цяло положително число, липсващо в масива. Оценете сложността на предложения от Вас алгоритъм в най-лошия случай.

Задача 5. Масивът $A\big[1\,\ldots\,n\big]$ съдържа n цели положителни числа — хонорари за n на брой дейности, всяка от които отнема една седмица. Предложете алгоритъм, който за време $o\left(n^2\right)$ намира съвкупност от дейности, осигуряваща желан паричен минимум S за най-малко седмици.

Задача 6. Алгоритъмът по-долу получава масив A, който съдържа n числа:

InversionSort($A\big[1\,\ldots\,n\big]$)
```
1  i ← 1
2  while i < n do
3      if A[i] > A[i+1]
4          swap(A[i], A[i+1])
5          if i = 1
6              i ← i + 1
7          else i ← i − 1
8      else i ← i + 1
```
а) Докажете, че алгоритъмът сортира масива $A\big[1\,\ldots\,n\big]$.
б) Ако времевата сложност на алгоритъма е $T(n)$, докажете, че $T(n) = \Theta\Big(n^2\Big)$.

Упътване: В подусловие "а" използвайте, че ако масивът A не е сортиран, то съществува индекс $i < n$, за който $A\big[i\big] > A\big[i+1\big]$.

РЕШЕНИЯ

Задача 1. Петте функции се подреждат по асимптотичен порядък на нарастване така:

$$\sqrt{n^7} \quad \prec \quad n^7 \log(8n+14) \quad \prec \quad 9\,.\,5^n \quad \prec \quad (5n)^{7n} \quad \prec \quad 7\,.\,5^{n^2}\,.$$

Доказателство: Първото неравенство се доказва с помощта на граница:

$$\lim_{n\to\infty} \frac{\sqrt{n^7}}{n^7 \log(8n+14)} = \lim_{n\to\infty} \frac{n^{3,5}}{n^7 \log(8n+14)} = \lim_{n\to\infty} \frac{1}{n^{3,5} \log(8n+14)} = \frac{1}{\infty} = 0\,.$$

Останалите три неравенства се доказват чрез логаритмуване:

$$\log\Big(n^7 \log(8n+14)\Big) = 7 \log n + \log\log(8n+14) \asymp \log n\,;$$

$$\log\Big(9\,.\,5^n\Big) = \log\big(5^n\big) + \log 9 = n \log 5 + \log 9 \asymp n\,;$$

$$\log\Big((5n)^{7n}\Big) = 7n \log(5n) = 7n \log 5 + 7n \log n \asymp n \log n\,;$$

$$\log\Big(7\,.\,5^{n^2}\Big) = \log\Big(5^{n^2}\Big) + \log 7 = n^2 \log 5 + \log 7 \asymp n^2\,.$$

(При опростяването пренебрегнахме константните множители и събираемите от нисък порядък.)

Понеже логаритмуването смалява разликите, то остава да докажем, че

$$\log n \quad \prec \quad n \quad \prec \quad n \log n \quad \prec \quad n^2.$$

От тези три неравенства:
— първото се доказва чрез граници (както по-горе), като се използва и правилото на Лопитал;
— третото се свежда до първото чрез деление на n;
— второто след деление на n се свежда до $1 \prec \log n$, което се доказва пак чрез граници.

Задача 2. Всяко рекурентно уравнение е от различен тип и се решава по съответния метод.

а) Това е линейно-рекурентно уравнение. Решава се чрез характеристично уравнение.
Отговор: $T(n) = C_1\,.\,2^n + C_2\,.\,8^n + C_3\,.\,n\,.\,8^n = \Theta\Big(n\,.\,8^n\Big)$.

б) И това е линейно-рекурентно уравнение, но свободният член е от дробна степен ($81/2 = 40{,}5$), затова то не може да се реши чрез характеристично уравнение. Решава се чрез развиване.
Отговор: $T(n) = T(0) + \sum\limits_{k=1}^{n} k^{40,5} = \Theta\Big(n^{41,5}\Big)$.

в) Това също е линейно-рекурентно уравнение, обаче с променлива дължина на историята. Заместваме n с $n+1$, от полученото уравнение вадим оригиналното и получаваме линейно-рекурентно уравнение с фиксирана дължина на историята: $T(n+1) = 2\,T(n) + 5\,.\,6^n$. То на свой ред се решава чрез характеристично уравнение.
Отговор: $T(n) = C_1\,.\,2^n + C_2\,.\,6^n = \Theta\Big(6^n\Big)$.

г) Решава се с помощта на мастър-теоремата: $k = \log_3 243 = 5$; $9n^4 = o\Big(n^{k-\epsilon}\Big) = o\Big(n^{5-\epsilon}\Big)$ например за $\epsilon = 0,5$ (защото $5 - 0,5 = 4,5 > 4$).
Отговор: $T(n) = \Theta\Big(n^5\Big)$.

Задача 3. Да означим с s_k стойността на променливата s в момента, когато се изпълнява проверката за край на цикъла на ред № 2, след като тялото му се е изпълнило k пъти $(k \geq 0)$.

Следователно $s_0 = 4$ (началната стойност от ред № 1).

От ред № 3 следва, че при $k > 0$ важи линейно-рекурентното уравнение $s_k = 8s_{k-1} + 6$. Решаваме го чрез характеристично уравнение и намираме $s_k = C_1 \,.\, 8^k + C_2$.

От $s_0 = 4$ и от рекурентното уравнение пресмятаме $s_1 = 38$. Във формулата с C_1 и C_2 заместваме $k = 0$ и $s_0 = 4$, после $k = 1$ и $s_1 = 38$. За двата неопределени коефициента се получава система от две линейни уравнения:

$$\left| \begin{array}{rcl} C_1 + C_2 & = & 4 \\ 8\,C_1 + C_2 & = & 38 \end{array} \right.$$

От системата намираме $C_1 = \dfrac{34}{7}$ и $C_2 = -\,\dfrac{6}{7}$. Следователно $s_k = \dfrac{34 \,.\, 8^k - 6}{7}$.

Решаваме относно k неравенството за край на цикъла:

$$s_k \;\leq\; n \quad\Longleftrightarrow\quad \frac{34 \,.\, 8^k - 6}{7} \;\leq\; n \quad\Longleftrightarrow\quad 8^k \;\leq\; \frac{7n+6}{34} \quad\Longleftrightarrow\quad k \;\leq\; \log_8 \left(\frac{7n+6}{34} \right).$$

Последното неравенство е равносилно на $k \;\leq\; \left\lfloor \log_8 \left(\dfrac{7n+6}{34} \right) \right\rfloor$, понеже k е цяло число.

Докато е в сила това неравенство, се изпълнява и цикълът. Алгоритъмът излиза от цикъла при първото k, нарушаващо неравенството, т.е. при $k \;=\; \left\lfloor \log_8 \left(\dfrac{7n+6}{34} \right) \right\rfloor + 1$.

а) Алгоритъмът връща стойност s_k, където k има стойността, посочена на предния ред. Тоест $s_k \;=\; \dfrac{34 \,.\, 8^k - 6}{7} \;=\; \dfrac{1}{7} \cdot \left(-\,6 + 34 \,.\, 8^{\left\lfloor \log_8 \left(\frac{7n+6}{34} \right) \right\rfloor + 1} \right)$ е върнатата стойност.

б) Тъй като всяка итерация на цикъла се изпълнява за константно време, то времевата сложност $T(n)$ на алгоритъма се определя от броя на итерациите, затова е равна по порядък на онази стойност на k, с която алгоритъмът приключва работа. Понеже се интересуваме само от порядъка на сложността, пренебрегваме константните събираеми и множители, основата на логаритъма и функцията "цяла част" и получаваме логаритмична сложност: $T(n) = \Theta(\log n)$.

Задача 5. "Най-малко седмици" ще рече "най-малък брой събираеми". За да получим желания паричен минимум S с най-малък брой събираеми, трябва да вземем най-големите елементи на масива. Следователно е добре да ползваме някоя бърза сортировка (HeapSort или MergeSort). После, започвайки от най-големите хонорари, натрупваме парични суми: $A[n]$, $A[n] + A[n-1]$, $A[n] + A[n-1] + A[n-2]$ и т.н. Тоест, ако $A[n] \geq S$, избираме само последната дейност. Ако $A[n] < S$, но $A[n] + A[n-1] \geq S$, избираме само последните две дейности. И така нататък. Спираме веднага щом поредният сбор стане по-голям или равен на S, тоест спираме при най-малкото k, за което $A[n] + A[n-1] + A[n-2] + A[n-3] + \ldots + A[n-k+1] + A[n-k] \geq S$, и избираме да извършим дейностите с поредни номера n, $n-1$, $n-2$, $\ldots$, $n-k$ (след сортирането, т.е. най-скъпо платените дейности).

Спираме също и ако изчерпим масива, т.е. ако $A[n] + A[n-1] + \ldots + A[3] + A[2] + A[1] < S$. В този случай желаната парична сума S не може да се образува: липсват достатъчно хонорари.

Сортирането изисква време $\Theta(n \log n)$. Образуването на сумата S изразходва време $\Theta(n)$. Общото време е $\Theta(n \log n) + \Theta(n) = \Theta(n \log n) = o\left(n^2\right)$, т.е. алгоритъмът е достатъчно бърз.

Задача 4. Очевидно отговорът е някое от числата 1, 2, 3, ... , n+1.

Наивният алгоритъм проверява първо дали масивът съдържа 1, после 2 и т.н. до n. Всяка проверка изисква линейно време. В най-лошия случай (когато масивът съдържа всички числа от 1 до n) се правят n проверки. Следователно сложността на този алгоритъм е $\Theta(n^2)$, което не е най-бързо.

Друго решение е да сортираме масива за време $\Theta(n \cdot \log n)$, а после да търсим първото липсващо число за линейно време:

```
SmallestMissing(A[1...n]: array of integers): integer
Sort(A) // HeapSort или MergeSort
if A[1] > 1
  return 1
for k ← 1 to n-1
  if A[k+1] - A[k] > 1
    return A[k] + 1
return A[n] + 1
```

Времевата сложност на този алгоритъм е $\Theta(n \cdot \log n) + \Theta(n) = \Theta(n \cdot \log n)$, което не е най-бързо. Не можем да ползваме директно (т.е. без промени) специалните сортировки CountingSort и RadixSort, защото не са изпълнени условията, при които тези сортировки могат да се прилагат. Например оригиналният вариант на CountingSort е неизползваем, тъй като в условието е казано, че числата в масива A могат да са много големи.

Като вземем предвид, че първото липсващо число е от 1 до n+1, можем да приложим следната *модификация* на CountingSort:

```
SmallestMissing(A[1...n]: array of integers): integer
C[1...n]: array of bool
for k ← 1 to n
  C[k] ← false
for k ← 1 to n
  if A[k] ≤ n
    C[A[k]] ← true
for k ← 1 to n
  if C[k] = false
    return k
return n+1
```

Този алгоритъм работи в линейно време $\Theta(n)$ и е най-бързият възможен, поне по порядък (не е изключено да има друг, също толкова бърз алгоритъм). Твърдението за оптималност следва от това, че всеки алгоритъм трябва да извърши поне n операции по прочитане на входните данни (ако например числото 1 липсва в масива, алгоритъмът няма как да установи това, освен като изследва всички елементи на масива).

Задача 6. Ще ползваме очевидната инварианта: Всеки път, когато алгоритъмът е на ред № 2, подмасивът $A\big[\,1\,\ldots\,i\,\big]$ е подреден (не съдържа инверсии).

Интересен е въпросът дали i достига стойност n. Ако това стане, масивът ще бъде сортиран.

Ще докажем по индукция лема, която дава утвърдителен отговор на горния въпрос и влече коректността на алгоритъма.

Лема: За всяко $k \in \Big\{\ 1,\ 2,\ 3,\ \ldots\ ,\ n\ \Big\}$ променливата i достига стойност k при работата на алгоритъма $InversionSort$.

Доказателство: i достига стойност 1 още при първото изпълнение на ред № 2.

Нека i достига стойност k на ред № 2. Има две възможности — неравенството на ред № 3 или е изпълнено, или не е.

Ако неравенството на ред № 3 е в сила, то елементът $A\big[k+1\big]$ след най-много k размени ще бъде поставен на точното си място в подмасива $A\big[\,1\,\ldots\,k+1\,\big]$ и след още най-много k преминавания през цикъла променливата i ще достигне стойност $k+1$.

В другия случай (когато неравенството на ред № 3 не е в сила) променливата i веднага достига стойност $k+1$.

Прилагаме принципа на математическата индукция и завършваме доказателството на лемата. ■

Сега да означим с t_k броя на изпълненията на тялото на цикъла $while$ до момента, когато променливата i за пръв път достига стойност k. От доказателството на лемата се вижда, че от момента t_k до момента t_{k+1} тялото на цикъла се изпълнява не повече от $2k$ пъти, т.е. важи рекурентното неравенство $t_{k+1} \le t_k + 2k$. След развиване на неравенството се получава $t_n \ \le\ t_{n-1} + 2(n-1) \ \le\ t_{n-2} + 2(n-2) + 2(n-1) \ \le\ \ldots \ \le\ t_1 + 2\,.\,\Big(1 + 2 + \ldots + (n-1)\Big)$, т.е. $t_n \ \le\ t_1 + (n-1)\,n$. Понеже $t_1 = 0$, то $t_n \ \le\ (n-1)\,n$. Алгоритъмът завършва, когато променливата i достигне стойност n, затова неговата времева сложност $T(n) \asymp t_n$. Следователно $T(n) \preceq (n-1)\,n$, т.е. $T(n) \preceq n^2$.

Остава да забележим, че при всяко преминаване през цикъла алгоритъмът унищожава най-много една инверсия (той унищожава инверсия от вида $< i,\ i+1 >$, която не влияе на останалите инверсии в масива). Ако подадем на входа обратно нареден масив, то броят на инверсиите в него ще бъде $(n-1)\,n\,/\,2$, следователно $T(n) \succeq (n-1)\,n\,/\,2$, т.е. $T(n) \succeq n^2$.

От двете асимптотични неравенства $T(n) \preceq n^2$ и $T(n) \succeq n^2$ следва, че $T(n) \asymp n^2$.

Окончателно, алгоритъмът има квадратична времева сложност: $T(n) = \Theta\Big(n^2\Big)$.