Семестриално контролно по "Дизайн и анализ на алгоритми" (СУ, ФМИ, 23. 04. 2016 г.)

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

Задача	1	2	3	4	5	6	Общо
получени точки							
максимум точки	20	5+5+10	20+20	20	50	20	170

Забележка: За отлична оценка са достатъчни 100 точки!

Задача 1. Задачата за търговския пътник може да се реши по различни начини:
— Пълното изчерпване има времева сложност $\Theta(n!)$, където n е броят на градовете.
— Алгоритъмът на Хелд—Карп (с динамично програмиране) има сложност $\Theta\left(n^2\, 2^n\right)$.
— Алгоритъмът на Гебауер (за графи с върхове от степен до 4) има сложност $\mathrm{O}\left(n^c\, 1{,}733^n\right)$ за някакво c.
Кой от алгоритмите е най-бърз и кой е най-бавен? Отговорът да се обоснове!

Задача 2. Решете рекурентните уравнения с точност до порядък:

а) $T(n) = T(n-1) + \sqrt[7]{n^{25}}$; б) $T(n) = 625\, T\left(\frac{n}{5}\right) + 6n^3$; в) $T(n) = \sum_{k=1}^{n-1} k\, .\, T(k)$.

Задача 3. Разглеждаме следния алгоритъм:

```
f (n: unsigned integer)
1  a ← 4
2  for k ← 1 to n
3      a ← 3 * a + 2
4  return a
```

а) Намерете явна формула за върнатата стойност $f(n)$ за всяко цяло $n \geq 0$.
б) Предложете по-бърз алгоритъм за смятане на $f(n)$. Сравнете по порядък времевите сложности на двата алгоритъма.

Задача 4. Имаме кутия с 42 жълти и 53 сини пионки и разполагаме с неограничен запас от жълти пионки (извън кутията). Започваме следната игра:

```
1  while има поне две пионки в кутията do
2      наслуки избираме две пионки от кутията;
3      if двете пионки са с еднакъв цвят,
4          изваждаме двете пионки и добавяме в кутията една от запасните жълти пионки;
5      else // (избрани са една жълта и една синя пионка)
6          изваждаме жълтата пионка, а синята оставяме в кутията.
```

Играта завършва, когато в кутията остане една пионка. Какъв ще бъде цветът ѝ?
Обосновете подробно отговора си с помощта на инварианта на цикъла.

Задача 5. Получили сме n поръчки, всяка от които носи приблизително една и съща печалба. Изпълнението на i-тата поръчка изисква a_i дена, а ние имаме общо D дена (след изтичането на които някакъв необходим ресурс вече няма да бъде на наше разположение). Предложете алгоритъм, който по дадени цели положителни числа D и $a_1, a_2, \ldots, a_n$ отпечатва списък от максималния възможен брой поръчки, които можем да изпълним за D дена, ако във всеки момент можем да се занимаваме само с една поръчка. Алгоритъм с максимална времева сложност $\mathrm{O}(n)$ носи 50 точки, а със сложност $\mathrm{O}(n \log n)$ — 20 точки. По-бавен алгоритъм не носи точки.
Анализирайте времевата сложност на предложения от Вас алгоритъм.

Задача 6. Турнир по тенис с 16 играча се провежда по системата на елиминациите – след всеки мач победеният играч отпада. След излъчване на победителя се играят няколко допълнителни мача, за да се определи втория по сила. Какъв е минималният общ брой мачове в турнира и по какъв начин трябва да се изиграят?

РЕШЕНИЯ

Задача 1. От трите предложени алгоритъма пълното изчерпване е най-бавно, а алгоритъмът на Гебауер е най-бърз. Това следва от неравенствата

$$n^c\,1{,}733^n \prec n^2\,2^n \prec n!$$

(където всички сравнения са по асимптотичен порядък на нарастване).

Първото неравенство е равносилно на

$$n^{c-2} \prec \left(\frac{2}{1{,}733}\right)^n.$$

В условието не е уточнена стойността на константата c, затова разглеждаме два случая.

Ако $c \le 2$, лявата страна клони към 0 или 1, а дясната — към безкрайност $\left(\frac{2}{1{,}733} > 1\right)$. В такъв случай неравенството следва от границата

$$\lim_{n\to\infty} \frac{n^{c-2}}{\left(\frac{2}{1{,}733}\right)^n} = \frac{0 \text{ или } 1}{\infty} = 0.$$

Ако $c > 2$, то двете страни на неравенството клонят към безкрайност. В този случай неравенството се доказва чрез логаритмуване:

$$\log\left(n^{c-2}\right) \asymp (c-2)\log n \asymp \log n\,, \qquad \log\left(\left(\frac{2}{1{,}733}\right)^n\right) \asymp n \log\left(\frac{2}{1{,}733}\right) \asymp n\,,$$

като се вземе под внимание, че $\log n \prec n$. Това на свой ред следва от границата

$$\lim_{n\to\infty} \frac{\ln n}{n} = \left[\frac{\infty}{\infty}\right] = \lim_{n\to\infty} \frac{1/n}{1} = 0.$$

Да отбележим, че първото неравенство в оригиналния си вид, т.е.

$$n^c\,1{,}733^n \prec n^2\,2^n,$$

не може да се докаже чрез логаритмуване направо (без предварително преобразуване):

$\log\left(n^c\,1{,}733^n\right) \asymp c\log n + n\log 1{,}733 \asymp \log n + n \asymp n\,;$

$\log\left(n^2\,2^n\right) \asymp 2\log n + n\log 2 \asymp \log n + n \asymp n\,;$

защото се получава равенство на логаритмите по порядък: $n \asymp n$, а от него не следва нищо.

Второто неравенство, т.е. $n^2\,2^n \prec n!$, може да се докаже чрез логаритмуване:

$$\log\left(n^2\,2^n\right) \asymp 2\log n + n\log 2 \asymp \log n + n \asymp n\,, \qquad \log(n!) \asymp n\log n\,.$$

Тъй като $n \prec n\log n$ и $n\log n \to \infty$ при $n \to \infty$, то $n^2\,2^n \prec n!$ (антилогаритмуването усилва разликите). Неравенството $n \prec n\log n$ следва от границата

$\lim\limits_{n\to\infty} \dfrac{\cancel{n}\log n}{\cancel{n}} = \lim\limits_{n\to\infty} \log n = \infty$ (логаритъмът е при произволна основа, по-голяма от 1).

Задача 2. а) Развиваме уравнението: $T(n) = T(0) + 1^{25/7} + 2^{25/7} + \ldots + n^{25/7} \asymp n^{32/7}$.
б) С мастър-теоремата: $k = \log_5 625 = 4$, $n^{k-\varepsilon} \succ 6n^3$, напр. $\varepsilon = 0{,}01$. Значи, $T(n) = \Theta\left(n^4\right)$.
в) $T(n) = \sum\limits_{k=1}^{n-1} k\,.\,T(k)$. Заместваме n със $n-1$: $T(n-1) = \sum\limits_{k=1}^{n-2} k\,.\,T(k)$. Това уравнение го вадим от оригиналното: $T(n) - T(n-1) = (n-1)\,.\,T(n-1)$, тоест $T(n) = n\,.\,T(n-1)$.
Развиваме полученото уравнение: $T(n) = n(n-1)\,.\,T(n-2) = \ldots = n!\,T(0) = \Theta(n!)$.

Задача 3. Да означим с a_k стойността на променливата a след k-тата итерация на цикъла (a_0 е началната стойност). Трасираме програмния код и получаваме първите няколко стойности от редицата a_k:

k	0	1	2	3	4	5	6
a_k	4	14	44	134	404	1214	3644

Както се вижда от инструкцията в тялото на цикъла, тази редица удовлетворява нехомогенното линейно-рекурентно уравнение $a_k = 3a_{k-1} + 2$. С помощта на характеристично уравнение намираме $a_k = C_1 . 3^k + C_2$. От $a_0 = 4$ и $a_1 = 14$ се получава системата

$$\left| \begin{array}{rcl} C_1 + C_2 & = & 4 \\ 3C_1 + C_2 & = & 14 \end{array} \right.$$

с единствено решение $C_1 = 5,\ C_2 = -1$. Затова $a_k = 5 . 3^k - 1$ за всяко цяло $k \geq 0$.

От условието за край на цикъла се вижда, че функцията f връща стойността $f(n) = a_n = 5 . 3^n - 1$.

Даденият алгоритъм има времева сложност $\Theta(n)$ заради цикъла на редове №2 – №3. Алгоритъмът може да се ускори до $\Theta(\log\ n)$, ако формулата $f(n) = 5 . 3^n - 1$ се програмира чрез бързо степенуване, което по същество се свежда до последователно повдигане на квадрат. Новият вариант е по-бърз, защото $\log n \prec n$, което следва от границата

$$\lim_{n\to\infty} \frac{\ln n}{n} = \left[\frac{\infty}{\infty}\right] = \lim_{n\to\infty} \frac{1/n}{1} = 0.$$

Задача 4 се решава с помощта на следната инварианта на цикъла:
Всеки път, когато се изпълнява проверката за край на цикъла на ред № 1 от алгоритъма, в кутията има нечетен брой сини пионки.

Доказателство на инвариантата:

База: Първото изпълнение на проверката за край на цикъла е в началото на играта. Тогава в кутията има 53 сини пионки. Тъй като 53 е нечетно число, то инвариантата е в сила.

Поддръжка: Нека след няколко хода (преди края на играта) в кутията има нечетен брой сини пионки. Ще докажем, че след още един ход в кутията пак ще има нечетен брой сини пионки.

Действително, ако на следващия ход се случи да бъдат избрани една жълта и една синя пионка, то от ред № 5 и ред № 6 на алгоритъма следва, че броят на сините пионки в кутията няма да се промени. Ако пък се случи да бъдат избрани две жълти или две сини пионки, то от ред № 5 и ред № 6 на алгоритъма следва, че броят на сините пионки няма да се промени или ще намалее с две съответно.

И така, след всеки ход броят на сините пионки или намалява с две, или не се променя. Затова, ако след някой ход броят на сините пионки е нечетно число, след още един ход този брой пак ще бъде нечетно число.

С това инвариантата е доказана.

Завършек на цикъла:

Отначало в кутията има $42 + 53 = 95$ пионки. Тъй като общият брой на пионките в кутията намалява с единица след всеки ход, то играта ще завърши точно след 94 хода. Тогава в кутията ще остане една пионка — жълта или синя; а броят на сините пионки в кутията ще бъде 0 или 1 съответно. Според доказаната инварианта този брой трябва да бъде нечетно число, т.е. 1. Следователно в края на играта в кутията ще остане една синя пионка.

Задача 5. Максимален брой поръчки ще изпълним, ако изберем изискващите най-малко дни.

Първи алгоритъм:
1) Сортираме поръчките по брой необходими дни
   (с някой от бързите алгоритми за сортиране, напр. HeapSort).
2) Изпълняваме поръчките последователно, започвайки от най-късата,
   докато изпълним всички поръчки или докато изтече времето D.

Анализ на алгоритъма: Първата стъпка (сортирането) има времева сложност $\Theta(n \log n)$.
Втората стъпка (обхождането на сортирания масив) има времева сложност $\Theta(n)$.
Общата времева сложност на алгоритъма е равна на $\Theta(n \log n) + \Theta(n) = \Theta(n \log n)$.

Втори алгоритъм:
1) Намираме медианата m на масива $a_1, a_2, \ldots, a_n$ с помощта на алгоритъма PICK.
2) Разделяме масива с поръчките на две равни по големина половини:
   къси поръчки $(a_i < m)$ и дълги поръчки $(a_i > m)$.
   Забележка 1: Ако има поръчки с дължина m, някои от тях може да отидат в едната половина,
   някои — в другата, та двете половини да са равни по големина.
   Забележка 2: Ако n е нечетно число, то двете половини ще се различават с една единица.
3) Намираме сбора S на дължините на късите поръчки.
4) Ако $D = S$, изпълняваме всички къси поръчки и никоя от дългите.
5) Ако $D < S$, то времето, с което разполагаме, не стига дори само за късите поръчки. Затова:
   а) отхвърляме всички дълги поръчки;
   б) извикваме рекурсивно алгоритъма върху масива с късите поръчки и със същия срок D.
6) Ако $D > S$, то времето, с което разполагаме, е достатъчно за късите поръчки. Затова:
   а) изпълняваме всички къси поръчки за общо S дена; остават ни още $D - S$ дена;
   б) извикваме рекурсивно алгоритъма върху масива с дългите поръчки с $D - S$ вместо D.

Анализ на алгоритъма: Всички стъпки без № 5б и № 6б имат времева сложност $\Theta(n)$.
Стъпките № 5б и № 6б (рекурсивните извиквания) имат времева сложност $T\left(\frac{n}{2}\right)$.
Общата времева сложност на алгоритъма е равна на $T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$.
Коефициентът пред $T\left(\frac{n}{2}\right)$ е единица, защото от двете рекурсивни извиквания се изпълнява в най-лошия случай едното. Полученото рекурентно уравнение се решава чрез мастър-теоремата: $k = \log_2 1 = 0$, $n^{k+\varepsilon} \prec n$ например за $\varepsilon = 0,01$, следователно $T(n) = \Theta(n)$.

Задача 6. На турнира съпоставяме кореново двоично дърво: листата съответстват на играчите; разклоненията съответстват на срещите — ако върховете y и z са наследници на x, то y и z са играли мач и x е победителят (някой от играчите y и z); коренът съответства на шампиона. Схемата на провеждане на турнира съответства на структурата на дървото (тя може да бъде избрана от организаторите на турнира). Попълването на дървото със стойности за върховете, които не са листа, зависи от изходите на мачовете, т.е. от играчите, а не от организаторите.

Както и да бъде проведен турнирът, са нужни 15 мача за определяне на шампиона: трябва да бъдат елиминирани 15 играчи, а след всеки мач отпада един претендент.

Нека k е броят на играчите, провели мач с шампиона. Вторият по сила играч е непременно някой от тези k играчи, тъй като той може да е бил елиминиран само от шампиона. За да бъде излъчен вицешампион, е нужен втори, "малък" турнир за тези k играчи, т.е. още $k - 1$ мача.

Тъй като k е дължината на пътя от корена до листото, съответстващо на шампиона, то в най-лошия случай k е дължината на най-дългия път от корен до листо. За двоично дърво броят на листата не надхвърля 2^k, затова $16 \leq 2^k$, откъдето $k \geq 4$. Минимумът $k = 4$ се достига при пълно двоично дърво (8 осмифинала, 4 четвъртфинала, 2 полуфинала и 1 финал). Минималният брой мачове в най-лошия случай е 18: от тях 15 са за определянето на шампиона и $k - 1 = 3$ — за вицешампиона.