Зад. 1   Майкъл Зарков, 45655

а) Алгоритъм 1

На ред 1 "int i, s=2, m=n*n;" се изпълнява за константно време C_1.

Цикълът на ред 2 се изпълнява точно $m*n = n*n*n = n^3$ пъти и тялото му се изпълнява за константно време C_2.

Ред 4 - константно време C_3.

Получаваме $C_1 + C_2 n^3 + C_3 \asymp n^3$.

б) Алгоритъм 2

Ред 1 и 2 се изпълняват за константно време C_1. За $n \geq 10$ не влизаме в if-а на ред 1 (и съответно не излизаме от ф-ята).

Условието на цикъла на ред 3 никога не е изпълнено, защото на ред 2 инициализираме $j=6$. Следователно редове 3, 4, 5 се изпълняват за константно време C_2.

На ред 6 за $n \geq 8$ условието на цикъла е изпълнено. На ред 8 викаме рекурсивно $g(n-1)$ и $g(n-2)$. Излизаме от цикъла, защото на ред 7 $j=n$ и $n-n=0 \neq 1$.

На ред 9 тялото на цикъла се изпълнява само веднъж (първо $j=n \geq n$, после $j=2 \ngeq n$ за $n \geq 3$. Нямаме безкраен цикъл, защото още на ред едно искаме $n \geq 10$). Викаме $g(n-2)$.

Получаваме рекурентното у-ние: $T(n) = c + T(n-1) + T(n-2) + T(n-2)$

$T(n) = T(n-1) + 2T(n-2) + C$

Решаваме чрез метода на характеристичното уравнение:

$x^n = x^{n-1} + 2x^{n-2} \quad | : x^{n-2} \neq 0$

$x^2 - x - 2 = 0$

$x_1 = -1, \; x_2 = 2 \longrightarrow \{-1, 2\}_M \Big| \longrightarrow \{-1, 1, 2\}_M$

$C = n^0 1^n . C \longrightarrow \{1\}_M$

$\Rightarrow T(n) = A(-1)^n + B(1)^n + D2^n \simeq 2^n$, <u>отг: $T(n) \simeq 2^n$</u>

(A, B, D - константи)

б) <u>Алгоритъм 3</u>

Ред 1 и 2 - константно време

Ред 3 - рекурсивно викане $h(\frac{n}{3})$

Ред 4 и 5 - $\lg n$ на брой итерации за константно време.

Ред 6 - рекурсивно викане $h(\frac{n}{3})$

Ред 7 - константно

Получаваме уравнението: $T(n) = 2T(\frac{n}{3}) + \lg n$

Решаваме с Мастър теоремата:

$\lg n = O(n^k)$, където $k = \log_3^2 > 0 \overset{\text{М.Т.}}{\Rightarrow} T(n) \simeq n^k$

Зад. 2

а) Инверсният вектор на перутацията е:

(4, 1, 1, 0, 2, 3, 0, 0, 0)

б)

Дата / Date

```
Get-permutation (I[1...n]: валиден ин-
                 версен вектор )
1   P[1...n] ← 0
2
3   for i ← 1 to n
4       k ← 1
5       empty ← 0
6
7       while empty < I[i]
8           if P[k] = 0
9               empty ← empty + 1
10          k ← k+1
11
12      while P[k] ≠ 0
13          k ← k+1
14
15      P[k] ← i
16
17  return P[1...n]
```

По подаден валиден инверсен вектор $I[1...n]$ функцията Get-permutation връща съответната му перутация на числата $\{1...n\}$.

Алгоритъмът работи като слага числата $1, 2, ... n$ на точната им позиция в перутацията. Първо намира позицията на числото 1 и го слага накрая в $P[1...n]$. После слага числото 2, после 3 и т.н. до n.

Неформално обяснение/обосновка за коректността

На ред 1 заделяме нов масив P[1...n] и го инициализираме с нула на всяка позиция. В P[1...n] ще запишем перпутацията. Ако P[i] = 0, това означава, че P[i] е свободна позиция (можем да записваме в нея. Ако P[i] ≠ 0, няма да пишем в P[i].

Инвариантата за цикъла на ред 3:

При i-тото достигане на ред 3 има точно едно срещане на числата 1, 2 ... i-1 и тези числа са на общите позиции, както в перпутацията на инверсния вектор I[1...n]. Останалите елементи на P[1...n] са нули.

1. База: i = 1 инвариантата е изпълнена (P[1...n] съдържа само нули).

2. Поддръжка:

Нека инвариантата е вярна за някое достигане i на ред 3, което не е последно. (ИД)

На ред 4 имаме итератор k, който е поставен в началото на масива (k ← 1, ред 4). While цикълът на ред 7 отброява I[i] - на брой празни позиции от началото на P[1...n] (празните позиции ~~са~~ са с числото нула). Също така този цикъл пресмята непразните позиции, защото по ИД те са с числата {1...i-1} и са на правилните позиции, и не на по-голяма от i. На отбросените празни места със сигурност има числа > i, защото вляво по-малки (ИД) вече са сложени.

Цикълът на ред 12 върти докато стигнем до свободна позиция, след като сме срещнали I[i] на брой празни позиции.

След изпълнението на цикъла на ред 12 сме намерили търсената позиция на i. На ред 15 слагане i на тази позиция.

Дали наистина i е на правилното място в пермутацията?

Допуснем, че не е. Тогава:

1сл. i трябва да застане на място от от местата на числата $1 \dots i-1$.

Това е ↯ с ИИ ($1 \dots i-1$ са вече на местата си).

2сл. i трябва да е взето от определено му от алгоритъма място.

Тогава броят на празните места преди i ще е $< I[i]$, защото i ще е заело едно от тези места ↯ с броя на инверсиите.

3сл. i тр. да е късно.

Тогава празните места ще са повече, но нататък празни места могат да стоят само числа $> i$ (от ИИ $1 \dots i-1$ вече са сложени)

$\Rightarrow$ ↯ с броя на инверсиите $I[i]$.

$\Rightarrow$ алгоритъмът е сложил i на правилното му място в пермутацията. Тогава инвариантата е изпълнена и при $n+1$-вото достигане на ред 3 числата $\{1, 2 \dots ((n+1)-1)\} = \{1 \dots n\}$ са на местата си в пермутацията. Излизане от цикъла и на ред 17 връщане пермутацията. □

Асимптотика по време

$\forall m \in \{1 \dots n\}$ (ред 3) записването от 1 (ред 4, $k \leftarrow 1$) и с цикъла на ред 7 и ред 12 търсим мястото на m като увеличаване k с единица на всяка итерация от цикъла на ред 7 до ред 12.

Когато достигнем точно позицията на т в пермутацията (к = позицията на т в перму.) тогава на ред 15 записване т на мястото на a_k. Имаме константна работа в тялото на цикъла на ред 7 и 12. Позициите на числата a_k $1, 2, 3, \ldots n$

$\Rightarrow$ получаване ~~работа~~ $1+2+3+\ldots+n = \frac{n(n+1)}{2} \asymp n^2$