А Н А Л И З Н А А Л Г О Р И Т М И

КОНТРОЛНО № 1 ПО ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ

ЗА СПЕЦИАЛНОСТ “КОМПЮТЪРНИ НАУКИ” — СУ, ФМИ, 2. КУРС, 1. ПОТОК

(24 МАРТ 2022 Г.)

Задача 1. Анализирайте времевата сложност $T_1(n)$ на следния алгоритъм:

```
ALG_1(A[1...n])
if n < 1
    return 0
if n < 100
    return A[1]
k ← ⌊n/64⌋
p ← k × 32
r ← n - k
k ← k + 1
x ← A[p]
y ← ALG_1(A[1...k])
z ← ALG_1(A[r...n])
return x + y + z
```

Задача 2. Анализирайте времевата сложност $T_2(n)$ на следния алгоритъм:

```
ALG_2(n)
r ← 1
p ← 6
k ← p
while k ≤ n do
    print k
    k ← 16 × p - 60 × r
    r ← p
    p ← k
```

Задача 3. Разглеждаме следния алгоритъм:

```
ALG_3(A[1...n])
for k ← 2 to n do
    if A[k] < A[k-1]
        return k
return 1
```

Каква стойност връща алгоритъмът `ALG_3`? Докажете отговора си с подходящ инвариант и полуинвариант.

Във всяка от трите задачи параметърът n е цяло неотрицателно число.

СХЕМА ЗА ТОЧКУВАНЕ

Контролното носи 20 точки, разпределени по задачи, както следва:

Задача 1 носи 6 точки, от които: 3 точки за съставяне на рекурентно уравнение за времевата сложност и още 3 точки за решаване на уравнението.

Задача 2 носи 7 точки, в това число: 3 т. за съставяне на рекурентно уравнение за редицата от числа, отпечатани на изхода; 3 т. за решаване на уравнението; още 1 т. за намиране на времевата сложност на алгоритъма.

Задача 3 носи 7 точки: 2 т. за формулиране на верен и използваем инвариант; още 2 т. за неговото доказателство (1 т. за базата и 1 т. за индуктивната стъпка); друга 1 т. за доказателство, че алгоритъмът ще завърши; и 2 т. за извеждане на отговора (1 т. за формулиране на отговора и 1 т. за доказването му).

РЕШЕНИЯ

Задача 1. Алгоритъмът `ALG_1` се извиква рекурсивно два пъти, като всяко от двете извиквания е върху масив с дължина, равна на една шейсет и четвърта от дължината на входния масив. Останалата работа отнема константно време, тъй като алгоритъмът не съдържа цикли. Следователно за достатъчно големи n времевата сложност $T_1(n)$ на алгоритъма удовлетворява рекурентно уравнение от тип "разделяй и владей":

$$T_1(n) = 2\,T_1\left(\frac{n}{64}\right) + \Theta(1).$$

Пресмятаме $\log_{64} 2 = \frac{1}{6}$. От първия случай на мастър-теоремата намираме

$$T_1(n) = \Theta(n^{1/6}) = \Theta\left(\sqrt[6]{n}\right).$$

Задача 2. Трасираме алгоритъма `ALG_2` и виждаме, че отпечатва следната редица: 6, 36, 216 и тъй нататък. Забелязваме, че тези числа представляват последователните степени на шестицата, тоест алгоритъмът отпечатва числото $k_m = 6^m$ при m-тото изпълнение на тялото на цикъла. Ще докажем това.

Според псевдокода, в променливата p се пази предишната стойност на k, а в променливата r — по-предишната. Следователно

$$k_m = 16\,k_{m-1} - 60\,k_{m-2} \text{ за всяко цяло } m \geq 3.$$

От това линейно-рекурентно уравнение и началните условия $k_1 = 6$ и $k_2 = 36$ можем с характеристично уравнение да изведем желаната формула: $k_m = 6^m$. Или използваме математическа индукция, стига да сме налучкали отговора в началото на решението.

Условието за край на цикъла $k_m \leq n$ решаваме като неравенство спрямо m и намираме броя на повторенията на тялото на цикъла:

$$k_m \leq n \;\Leftrightarrow\; 6^m \leq n \;\Leftrightarrow\; m \leq \log_6 n.$$

Тялото на цикъла се изпълнява $\lfloor \log_6 n \rfloor$ пъти и толкова числа се отпечатват. Това е и времевата сложност на алгоритъма: $T_2(n) = \Theta(\log n)$.

Задача 3. Алгоритъмът `ALG_3` връща най-малкия индекс (от 2 до n вкл.), чийто елемент е по-малък от предходния елемент; алгоритъмът връща единица, ако няма такъв елемент.

Инвариант: Всеки път, когато се изпълнява проверката за край на цикъла, е в сила неравенството $A[q-1] \leq A[q]$ за всяко $q = 2, 3, 4, \ldots, k-1$; тоест важат неравенствата $A[1] \leq A[2] \leq A[3] \leq \ldots \leq A[k-2] \leq A[k-1]$.

Доказваме инварианта с математическа индукция по номера на проверката.

База: Първата проверка се изпълнява при влизане в цикъла. Тогава $k = 2$ и инвариантът е тривиално верен: променливата q с квантор за всеобщност притежава празно множество от допустими стойности (или, което е същото, веригата се състои от $k - 2 = 0$ неравенства, тоест тя е празна конюнкция).

Индуктивна стъпка: Нека $A[1] \le A[2] \le A[3] \le \ldots \le A[k-2] \le A[k-1]$ при някоя проверка за край на цикъла, която не е последна. Следователно $k \le n$ и започва ново изпълнение на тялото на цикъла.

Да допуснем, че $A[k-1] > A[k]$. В такъв случай алгоритъмът спира работа и връща k, което противоречи на предположението, че ще има поне още една проверка за край на цикъла. Ето защо допускането не е вярно.

Значи, вярно е обратното неравенство: $A[k-1] \le A[k]$. Като го добавим към веригата от неравенства, спомената в индуктивното предположение, я удължаваме с едно неравенство и тя приема вида:

$$A[1] \le A[2] \le A[3] \le \ldots \le A[k-2] \le A[k-1] \le A[k].$$

След това k се увеличава с единица и същата верига вече се записва така:

$$A[1] \le A[2] \le A[3] \le \ldots \le A[k-3] \le A[k-2] \le A[k-1].$$

Алгоритъмът преминава към новата проверка за край на цикъла. В този миг важи новата верига от неравенства, която съдържа едно неравенство повече, но поради увеличаването на k формалният запис е същият: $A[q-1] \le A[q]$ за всяко $q = 2, 3, 4, \ldots, k-1$; т.е. инвариантът важи и при новата проверка.

Дотук доказахме инварианта. С негова помощ ще обосновем отговора.

Завършек: Алгоритъмът ще завърши, защото се състои от единствен цикъл, който е цикъл по брояч. По-формално, полуинвариант е броячът k на цикъла: стойността на брояча нараства с единица след всяко изпълнение на тялото, затова от 2 до $n + 1$ ще достигне за $(n + 1) - 2 = n - 1$ изпълнения на тялото на цикъла. Цикълът завършва след най-много $n - 1$ изпълнения на тялото, когато k достигне $n + 1$ (или по-рано, ако бъде намерен подходящ елемент).

Първи случай: Цикълът завършва при някоя проверка на условието за край, т.е. при $k = n + 1$. В доказания инвариант заместваме $k = n + 1$ и получаваме:

$$A[1] \le A[2] \le A[3] \le \ldots \le A[n-1] \le A[n],$$

което означава, че няма елемент, по-малък от предходния. Точно такъв отговор (служебна стойност 1) връща алгоритъмът чрез оператора след края на цикъла.

Втори случай: Цикълът завършва предсрочно — с оператора `return k`. От проверката на предходния ред от псевдокода следва, че $A[k-1] > A[k]$. Съчетаваме това неравенство с инварианта от последната досега проверка на условието за край на цикъла и получаваме следната верига от неравенства:

$$A[1] \le A[2] \le A[3] \le \ldots \le A[k-2] \le A[k-1] > A[k],$$

тоест върнатата стойност k е най-малкият индекс, чийто елемент е по-малък от предходния елемент.