Домашно №1 по Дизайн и анализ на алгоритми на специалност Компютърни науки, 2 поток, летен семестър 2022/2023г.

Камен Димитров Младенов, Група 6

27 Март 2023

Задача 1. Доказателство

I. Ще докажем, че $n^{1+\lg n} \preceq T(n)$ с индукция по n. По дефиниция, това е същото като да докажем, че съществува положителна константа c_1 и положително n_0, такова че за всяко $n \geq n_0$, $T(n) \geq c_1 n^{1+\lg n}$.

Индуктивното предположение е, че

$$T\left(\frac{n}{2}\right) \geq c_1 \left(\frac{n}{2}\right)^{1+\lg \frac{n}{2}}$$

Тогава

$$\begin{aligned}
T(n) = n^2 T\left(\frac{n}{2}\right) + 1 &\geq n^2 . c_1 \left(\frac{n}{2}\right)^{1+\lg \frac{n}{2}} + 1 \\
&= n^2 c_1 \frac{n^{\lg n}}{2^{\lg n}} + 1 && // \ \lg \frac{n}{2} = \lg n - \lg 2 \\
&= c_1 n^{1+\lg n} \frac{n}{2^{\lg n}} + 1 \\
&= c_1 n^{1+\lg n} + 1 && // \ \frac{n}{2^{\lg n}} = \frac{n}{n} = 1 \\
&\geq c_1 n^{1+\lg n} \text{ за всеки избор на } c_1 > 0 \text{ и } n_0 > 0
\end{aligned}$$

II. Ще засилим твърдението и ще докажем, че $T(n) \preceq n^{1+\lg n} - 1$ с индукция по n, тоест че $\exists c_2 > 0 \ \exists n_0 > 0 \ \forall n \geq n_0 : T(n) \leq c_2 n^{1+\lg n} - 1$

Индуктивното предположение е, че

$$T\left(\frac{n}{2}\right) \leq c_2 \left(\frac{n}{2}\right)^{1+\lg \frac{n}{2}} - 1$$

И тогава

$$\begin{aligned}
T(n) = n^2 T\left(\frac{n}{2}\right) + 1 &\leq n^2 \left(c_2 \left(\frac{n}{2}\right)^{1+\lg \frac{n}{2}} - 1 \right) + 1 \\
&= c_2 n^{\lg n + 1} - n^2 + 1 && // \text{ От I.} \\
&= c_2 n^{\lg n + 1} - 1 - n^2 + 2 \\
&\leq c_2 n^{\lg n + 1} - 1 \text{ за всеки избор на } c_2 > 0 \text{ и } n_0 \geq \sqrt{2}
\end{aligned}$$

Задача 2. Доказателство

Функцията връща a^3, ако $a > 0$ и 0 иначе, където a е целочислено.

Инвариант на цикъла: За всяко достигане на ред 4, стойностите на променливите x, y, z са:

$$\begin{aligned}
x &= 6(i+1) \\
y &= 1 + \sum_{j=0}^{i} 6j = 1 + 3i(i+1) \\
z &= \sum_{j=0}^{i-1} 1 + 3j(j+1) = i + 3 \sum_{j=0}^{i-1} j(j+1) = i + 3 \left(\sum_{j=0}^{i-1} j^2 + \sum_{j=0}^{i-1} j \right) \\
&= i + 3 \left(\frac{(i-1)i(2(i-1)+1)}{6} + \frac{(i-1)i}{2} \right) = i + 3 \left(\frac{2i^3 - 2i^2}{6} \right) = i^3
\end{aligned}$$

База: При първото достигане на ред 4, $i = 0$ и променливите x, y, z имат стойностите, зададени на ред 2. В същото време, при инварианта $x = 6(0+1) = 6, y = 1+3.0.(0+1) = 1, y = \sum_{j=0}^{-1} 1+3j(j+1) = 0$.

Поддръжка: Допускаме, че твърдението е изпълнено за k-тото нефинално изпълнение на ред 4 (където $i = k - 1$). Разглеждаме следващото достигане на ред 4. i приема стойността k. В цикъла, след завършването на текущата итерация:

$$\begin{aligned}
&\text{От ред 5} \quad z = (k-1)^3 + (1 + 3(k-1)k) = k^3 - 3k^2 + 3k - 1 + 1 + 3k^2 - 3k = k^3 \\
&\text{От ред 6} \quad y = (1 + 3(k-1)k) + (6k) = 1 + 3k^2 - 3k + 6k = 1 + 3(k+1)k \\
&\text{От ред 7} \quad x = (6k) + 6 = 6(k+1)
\end{aligned}$$

В същото време, спрямо новото i и според твърдението, $x = 6(k+1), y = 1 + 3k(k+1), z = k^3$.

Терминация: От условието на ред 4, цикъла ще завърши при $(a+1)$-ото изпълнение, когато $i = a$. То тогава, от инварианта, върнатата стойност на ред 10 ще бъде точно $z = a^3$.

Задача 3. Решение

SomeAlg(A, l, h) сортира подмасива $A[l..h]$ на $A[1..n]$. Тоест, SomeAlg$(A, 1, n)$ ще сортира целия масив.

Ще докажем коректността на алгоритъма с индукция по $h-l$ (което се явява дължината на нашия подмасив минус едно).

База: $h - l <= 0$. В този случай, подмасивът или има само един елемент или е празен. И в двете ситуации, подмасивът е сортиран. Условието на ред 1 спира изпълнението на алгоритъма, без да е променил подмасива.

Индукционно предположение: Допускаме, че за някои l', h', SomeAlg(A, l'', h'') сортира подмасива $A[l''..h'']$ за всички $l'' \leq l', h'' \leq h'$.

Индукционна стъпка: Разглеждаме $h - l > h' - l'$. Редове 2 и 3 "сортират" първия и последния елемент в подмасива $A[l..h]$ помежду си. На ред 4, t става една трета от дължината на подмасива, закръглена надолу. Разглеждаме два случая относно условието на ред 5:

I. Ако $t = 0$, то дължината на подмасива може да бъде само 2. Тогава двата елемента ще бъдат двата "крайни" елемента и от редове 2 и 3, подмасивът ще бъде сортиран.

II. Иначе, ако подмасивът има повече от 2 елемента.
За удобство, обозначаваме $\alpha = A[l..(l+t-1)], \beta = A[(l+t)..(h-t-1)], \gamma = A[(h-t)..h]$ (всяка съседна трета от масива) и нека Γ да бъдат най-големите $(h-l)/3$ елемента (елементите които ще бъдат в γ след сортиране). От индукционно предположение, можем да сортираме всяка трета поотделно, както и съседни трети $(\alpha\beta, \beta\gamma)$.
В подмасива, елементите от Γ ще се съдържат в α, β и/или γ. На ред 6 сортираме $\alpha\beta$, което означава че всички елементи от Γ, които са били в $\alpha\beta$, сега се съдържат само в β. На ред 7 сортираме $\beta\gamma$, което означава че всички елементи от Γ се намират в γ и са сортирани. На ред 8 сортираме всички останали елементи, които са в $\alpha\beta$ (и са по-малки от тези в γ).
Така целия подмасив е вече сортиран.

Сложност по време: ако на алгоритъма са му подадени n елемента, в общия случай на редове 2, 3, 4 и 5 той извършва работа, която не зависи от входа, тоест константа работа c, докато на редове 6, 7 и 8, общо три пъти извиква себе си с $\frac{2}{3}$ дължината на входа. Тоест, рекурентното уравнение ще бъде:

$$T(n) = 3T\left(\frac{2}{3}n\right) + c$$

От 1ви случай на мастър теоремата:

$$a = 3, b = \frac{3}{2}, f(n) = c \preceq \frac{n^{\log_{\frac{3}{2}} 3}}{n^{\varepsilon}} \implies T(n) \asymp n^{\log_{\frac{3}{2}} 3}$$