

Методи за решаване на рекурентни уравнения - продължение

III Чрез Мастер теоремата (МТ)

Master theorem:

Нека $a \geq 1$, $b > 1$ са константи и нека $f(n)$ е положителна функция. Нека

$$T(n) = aT\left(\frac{n}{b}\right) + f(n).$$

Тогаво асимптотиката на $T(n)$ е следната:

① $k = \log_b a$, $f(n) \leq n^{k-\epsilon}$ за $\epsilon > 0$. Тогаво $T(n) = \Theta(n^k)$

② $k = \log_b a$, $f(n) \asymp n^k$. Тогаво $T(n) = \Theta(n^k \lg n)$,
 $T(n) = \Theta(f(n) \lg n)$

③ $k = \log_b a$, $f(n) \geq n^{k+\epsilon}$ за $\epsilon > 0$ и
 $a f\left(\frac{n}{b}\right) \leq c f(n)$ за $0 < c < 1$ и за всички достатъчно големи n .

↑
условие за регулярност

Тогаво $T(n) = \Theta(f(n))$

! Интуиция за МТ: и в трите случая на МТ сравняваме броя на листата на дървото на рекурсията с функцията $f(n)$.

Анализ на коректността на алгоритми

Анализ на коректността на итеративни алгоритми

① Коректност на изхода.

Доказваме с инварианта на цикъла.
предикат

Предикатът трябва:

- Да е верен при първото влизане в цикъла.
- Верността му да се запазва след всяко изпълнение на цикъла
- В момента на напускане на цикъла неговата верност трябва да бъде директно това, което искаме да докажем за алгоритъма.

② Дали алгоритъмът ще приключи.

Доказваме с полуинвариант.

величина,
която се
меня строго
монотонно

$$x \quad y$$

1. $z = 6, t = 9, p = 1$

2. $z = 6, t = 8, p = 6$

3. $z = 6^2, t = 4, p = 6$

4. $z = 6^4, t = 2, p = 6$

5. $z = 6^8, t = 1, p = 6$

6. $z = 6^8, t = 0, p = 6 \cdot 6^8 = 6^9$

Задача 1:

① Коректност на изхода:

$$\text{mystery}(x, y) = \begin{cases} x^y, & x \neq 0 \text{ или } y \neq 0 \\ 1, & x = 0 \text{ и } y = 0 \end{cases}$$

Доказателство с инварианта:

1 случай: $x \neq 0$ или $y \neq 0$

Инварианта: За всяка проверка за край на цикъла е изпълнено $x^y = z^t * p$.

База: При първото влизане в цикъла:

$$z = x,$$

$$t = y,$$

$$p = 1.$$

Следователно $x^y = z^t = z^t * 1 = z^t * p$. **ОК!**

Поддръжка: Допускаме, че инвариантата е изпълнена за някоя проверка за край на цикъла, която **не е последна**.

След като проверката не е последна, значи $t > 0$. Разглеждаме два случая:

1 сл.: t е четно.

Проверката ($\text{if } (t \% 2 == 0)$) връща истина. Тогава:

$$z' \text{ (новата стойност на } z) = z * z,$$

$$t' \text{ (новата стойност на } t) = t / 2.$$

Запазва ли се инвариантата? ($? x^y = z'^{t'} * p$)

$$z'^{t'} * p = (z * z)^{(t / 2)} * p = z^{(2 * t / 2)} * p = z^t * p.$$

Вярно ли е, че $x^y = z^t * p$? **Да, от допускането.**

2 сл.: t е нечетно.

Проверката ($\text{if } (t \% 2 == 0)$) връща лъжа. Тогава:

$$p' \text{ (новата стойност на } p) = p * z,$$

$$t' \text{ (новата стойност на } t) = t - 1.$$

Запазва ли се инвариантата? ($? x^y = z^{t'} * p'$)

$$z^{t'} * p' = z^{(t - 1)} * (p * z) = z^{(t - 1)} * z * p = z^t * p.$$

Вярно ли е, че $x^y = z^t * p$? **Да, от допускането.**

Следователно поддръжката е **ОК!**

Терминация: При последната проверка за край на цикъла $t = 0$.

$$\text{Тогава } x^y = z^t * p = z^{0 * p} = 1 * p = p.$$

След края на цикъла връщаме точно p . **ОК!**

2 случай: $x = 0$ и $y = 0$

Но тогава $t = y = 0$. Следователно тялото на цикъла няма да се изпълни нито веднъж. Следователно $p = 1$ не си променя стойността. Накрая връщаме точно p , което има стойност единица. **OK!**

Алгоритъмът ще приключи: Полуинвариант е променливата t .

Стойността на t се намалява след всяко изпълнение на тялото на цикъла (ако е нечетно - с единица, ако е четно - двойно).

Да допуснем, че алгоритъмът не приключва.

Тогава $t_1, t_2, t_3, t_4, \dots, t_n, \dots$ е безкрайна редица от последователни стойности на t , т. че $t_1 > t_2 > t_3 > t_4 > \dots > t_n > \dots$

Но тогава това е безкрайно намаляваща редица от естествени числа. Но такава не съществува. Противоречие.

Следователно алгоритъмът **mystery** завършва.

Задача 2: ("Тек сред цифрове")

Даден е масив с n цели числа, където n е нечетно. Всяко число се среща по два пъти с възникване на едно, което е уникално. Намерете това число.

Решение:

Първи опит: За всяко число проверяваме в масива дали има друго, равно на първото (с два вложени цикъла).

Това решение работи, но в ^{средния} най-лошия случай има сложност

$\Theta(n^2)$

\Rightarrow Търсим друго решение.

Втори опит: Сортираме масива и сравняваме само съседни елементи.

Сложността на този алгоритъм е $\underbrace{O(n \log n)}_{\text{сортиране}} + \underbrace{O(n)}_{\text{сравнения}} = \boxed{O(n \log n)}$

Дали съществува по-бърз алгоритъм?

Трети опит:

uniqueElement ($A[1..n]$: array of integers): integer

1. result $\leftarrow A[1]$
2. for $k \leftarrow 2$ to n
3. result \leftarrow result xor $A[k]$
4. return result

Тук използваме свойствата на изключващата дизюнкция xor ($++: '^'$), която в лугая се прилага побитово:

- $0 \text{ xor } 0 = 0$
- $0 \text{ xor } 1 = 1$
- $1 \text{ xor } 0 = 1$
- $1 \text{ xor } 1 = 0$

Операцията xor е асоциативна и комутативна, което интуитивно означава, че дали ще приложим алгоритъма в A масива $[1, 3, 4, 17, 3, 17, 1]$ или в B масива $[1, 1, 17, 17, 3, 3]$ е БЕЗ значение.

Като резултат еднаквите числа ще се съкратят и променливата result ще е точно единствената уникална стойност.

Очевидно сложността на uniqueElement е $\boxed{O(n)}$.

Задача 3: ("Произведения на останалите числа")

Даден е масив с n числа. Търси се нов масив с n числа, k -тото от които е равно на произведението на всички числа от входния масив без k -тото, $k \in [1, \dots, n]$.
Съставете алгоритъм, който НЕ използва деление.

Решение:

Първи опит: За всяко число пресмятаме произв. на другите \Rightarrow квадратичен алгоритъм.

Втори опит:

productOfOthers ($A[1..n]$: array of numbers): $B[1..n]$: array of numbers

1. $B[n] \leftarrow 1$
2. for $k \leftarrow n$ downto 2
3. $B[k-1] \leftarrow B[k] * A[k]$ // $B[k] = A[k+1] \dots A[n]$
4. fromLeft $\leftarrow A[1]$
5. for $k \leftarrow 2$ to n
6. $B[k] \leftarrow B[k] * \text{fromLeft}$ // $B[k] = A[1] \dots A[k-1]$
7. fromLeft $\leftarrow \text{fromLeft} * A[k]$ // $A[k+1] \dots A[n]$

Пример: $A = [2, 5, 3, 7, 4, 6]$ } Резултатът от първия цикъл
 $B = [2520, 504, 168, 24, 6, 1]$

$A = [2, 5, 3, 7, 4, 6]$ } Резултатът от втория цикъл
 $B = [2520, 504, 168, 24, 6, 1]$
 $\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
 $[2520, 1008, 1680, 720, 1260, 840]$

Сложността на productOfOthers е $\boxed{O(n)}$ - очевидно оптимална

Задача 4:

Алгоритъмът връща дали n е просто.

1. Коректност на изхода

Инварианта: За всяка проверка за край на цикъла е изпълнено, че n няма делители в интервала $[2, \dots, i-1]$.

База: При първото влизане в цикъла $i = 2$. Следователно $[2, \dots, i-1] = [2, \dots, 1] = []$ и n няма делители в интервала $[]$. ОК!

Поддръжка: Да допуснем, че инвариантата е изпълнена за някоя проверка за край на цикъла, която не е последна. Тоест n няма делители в интервала $[2, \dots, i-1]$ за някое i .

Щом проверката не е последна, условието $n \% i == 0$ вътре в тялото на цикъла връща лъжа, т.е. i не е делител на n . Следователно n няма делители в интервала $[2, \dots, i]$. След това (при следващата проверка за край на цикъла) i се увеличава с единица, откъдето получаваме, че n няма делители в интервала $[2, \dots, i-1]$. ОК!

Терминация: Изпълнението на цикъла може да приключи от две места:

1) От проверката на ред 11. Това означава, че условието в if-а на ред 11 връща истина. Следователно $i \geq 2$ е нетривиален делител на n . Следователно n не е просто. На ред 12 връщаме лъжа. ОК!

2) Условието за край на цикъла връща лъжа, т.е. $i > \text{temp} = \text{sqrt}(n)$.

От инвариантата знаем, че n няма делители в интервала $[2, \dots, i-1]$. Но $\text{sqrt}(n) < i$, следователно $\text{sqrt}(n) \leq i - 1$. Следователно n няма делители в интервала $[2, \dots, \text{sqrt}(n)]$ и от твърдение * следва, че n е просто. ОК!

Твърдение *: Ако $n \geq 0$ няма делители в интервала $[2, \dots, \text{sqrt}(n)]$, то n е просто.

Доказателство: Да допуснем противното, т.е. n няма делители в интервала $[2, \dots, \text{sqrt}(n)]$ и n не е просто. След като n не е просто, съществува $k \geq 2$, такова че $k \mid n$. Но n няма делители в интервала $[2, \dots, \text{sqrt}(n)]$, следователно $\text{sqrt}(n) < k$, т.е. $k = \text{sqrt}(n) + t$, $t > 0$. Но щом $k \mid n$, то $(n/k) \mid n$. Но n няма делители в интервала $[2, \dots, \text{sqrt}(n)]$. Следователно $\text{sqrt}(n) < (n/k)$, т.е. $(n/k) = \text{sqrt}(n) + q$, $q > 0$.

Но $n < (\text{sqrt}(n) + t) * (\text{sqrt}(n) + q) = k * (n/k) = n$, т.е. $n < n$. Невъзможно!

Противоречието се получи, защото допуснахме, че n не е просто.

Следователно, ако $n \geq 0$ няма делители в интервала $[2, \dots, \text{sqrt}(n)]$, то n е просто.

2. Завършек

Алгоритъмът ще завърши, защото се състои от единствен цикъл,

който е цикъл по брояч. По-формално, полуинвариант е броячът i на цикъла:

стойността на брояча нараства с единица след всяко изпълнение на тялото,

затова от 2 до $\text{sqrt}(n)$ ще достигне за $\text{sqrt}(n) - 1$ изпълнения на тялото

на цикъла. Цикълът завършва след най-много $\sqrt{n} - 1$ изпълнения на тялото, когато i достигне $\sqrt{n} + 1$ (или по-рано, ако бъде намерен делител на n).

Първи случай: Цикълът завършва при някоя проверка на условието за край,

т.е. при $i = \sqrt{n} + 1$. От доказаната инварианта следва, че **n е просто**, което е точно резултатът, който връщаме на ред 14. Алгоритъмът приключва.

Втори случай: Цикълът завършва предсрочно - с оператора **return false**. Условието на ред 11 е върнало истина, т.е. **n не е просто**, което е точно това, което връщаме на ред 12. Алгоритъмът приключва.

Задача 5: ("Чупливи ел. крушки")

Имаме електрически крушки, които се гупят, ако паднат от определена висотина (една и съща за всички). Имаме и една 100-етажна сграда. Как да намерим с минимален брой опити от кой етаж се гупят крушките? А с минимален брой ступени крушки?

Решение:

Двоично търсене (мин. брой опити) - пускаме крушката от 50-ти етаж, ако се гупи - пускаме втората от 25-ти етаж, ако не се гупи - пускаме втората от 75-ти и т.н.
 \Rightarrow Мин. брой опити = $\lceil \log_2 100 \rceil = 7$.

Линейно търсене (мин. брой ступени крушки = 1)