ДАА У.
(2021-12-03)

Припомняне:

- Стъпки за доказателство на итеративен аглоритъм:

1. Терминация – доказваме, че алгоритъма изобщо завършва;
2. Формулиране на инварианта за всеки цикъл;
3. Доказателство на инвариантите (индукция);
4. Терминираща фаза;
5. Проверка за нарушаване на входните данни. Т.е. не се присвояват нови стойности или не се преоразмеряват (за колекции) данните от входа.

Зад. 1. Определета какво връща алгоритъмът и докажете коректността му.

function $f_1(A[1, \ldots, n] : arr\ of\ int) : int$

1. $\quad p \leftarrow \max(A[1,2])$
2. $\quad q \leftarrow \min(A[1,2])$
3. $\quad$ **for** $i \leftarrow 3$ **to** n **do**
4. $\qquad$ **if** $A[i] > p$ **then**
5. $\qquad\quad q \leftarrow p$
6. $\qquad\quad p \leftarrow A[i]$
7. $\qquad$ **else if** $A[i] > q$ **then**
8. $\qquad\quad q \leftarrow A[i]$
9. $\quad$ **done**
10. $\quad$ **return** q

Твърдение: Функцията f_1 връща втория по големина елемент от масива A.

Началните данни на масива A не се променят никъде. Имаме for цикъл, в които никъде в тялото му не се намалява променливата i и следователно цикъла описва една монотонно растяща функция на i, която има праг n. След for цикъла извършваме константна работа. Следователно алгоритъмът със сигурност ще завърши.

Инварианта на for цикъла: При всяко достигане на ред 3, което не е последното, в променливата p ще имаме максималната стойност на масива от A[1, ..., i - 1], а q ще съдържа втората по големина стойност от същия масив.

База: за i = 2 твърдението е изпълнено, т.к. p = max(A[1, 2]) и q = min(A[1, 2]). Преди началото на for цикъла всичко е изпълнено за елементи с индекси до i. Взимаме $i + 1^{\text{-вия}}$ елемент. Ако той е по-голям от максималния p, тогава новия най-голям елемент става A[i + 1], а втория по-големина ще е предишния най-голям елемент, който запаметяваме в q.

Ако пък сме влезли в else if, то ще знаем, че p със сигурност си е останал най-големия елемент, т.к. A[i + 1] ≤ p и q присвоява втория по големина елемент, т.к. A[i] > q.

След последната стъпка на for цикъла сме разгледали всички елементи и преди $n + 1^{\text{-вата}}$ стъпка имаме p = max(A[1, ..., n]) и q = max(A[1, ..., n] \ max(A[1, ..., n])).

Зад. 2. Определета какво връща алгоритъмът и докажете коректността му.

function $f_2(a, b : non{-}negative\ int) : int$

1. $s \leftarrow 0$
2. $p \leftarrow a$
3. $r \leftarrow b$
4. **while** $r > 0$ **do**
5.   **if** $r \% 2 = 1$ **then**
6. $r \leftarrow r - 1$
7. $s \leftarrow s + p$
8. $r \leftarrow r/2$
9. $p \leftarrow 2 * p$
10. **done**
11. **return** s

Нека a = 5 и b = 18:

while t	0	1	2	3	4	5
s	0	0	10	10	10	90
p	5	10	20	40	80	160
r	18	9	4	2	1	0

Хипотеза: s + p * r = a * b

На a и b никъде не им се присвояват нови стойности. Променливата r никъде не я инкрементираме, а винаги я делим на две, като когато е нечетно число вадим 1-ца преди това от нея. В крайна сметка винаги я намаляваме, а условието в while цикъла е да се изпълнява докато r > 0. Следователно while цикъла описва монотонно намаляваща функция на r и рано или късно ще спре.

База: 0 + a * b = a * b.

Нека преди $k^{\text{-тото}}$ изпълнение на while цикъла, което не е последното: s + p * r = a * b.

Нека проверим дали след изпълнението на тази $k^{\text{-та}}$ стъпка ще се запази инвариантата.

Поддръжка: I сл. r–четно, II сл. r–нечетно.

I сл. r–четно ⇒ не влизаме в if и изпълняваме само двата реда след него (8 и 9) ⇒

$$\begin{cases} s \longrightarrow s \\ p \longrightarrow 2 * p \\ r \longrightarrow \frac{r}{2} \end{cases} \Rightarrow ab = s + pr = s + \cancel{2}p \,.\, \frac{r}{\cancel{2}} = s + pr$$

II сл. r–нечетно ⇒ влизаме в if и изпълняваме и редовете 6 и 7 преди 8 и 9 ⇒

$$\begin{cases} s \longrightarrow s + p \\ p \longrightarrow 2 * p \\ r \longrightarrow \frac{r-1}{2} \end{cases} \Rightarrow ab = s + pr = (s + p) + \cancel{2}p \,.\, \frac{r-1}{\cancel{2}} = s + p + pr - p = s + pr$$

И в двата случая хипотезата е вярна (инвариантата се запазва).

Терминация. При последното изпълнение на while цикъла, r ще е 1, след което ще стане 0 и ще разделим тази нула на 2, което пак е 0. Тогава $s + \underbrace{pr}_{=0} = ab$ и връщаме s, което е равно на произведението на аргументире a и b.

(Идеята на задачи 1 и 2 е да покаже, че инвариантата понякога е по-удобно да се запише словестно, а в други случаи е по-удобно да се запише като формула)

Зад. 3. Определета какво връша алгоритъма и докажете коректността му.

function $f_3(A[1, \ldots, n] : int) : int$

```
counter ← 1
current ← A[1]
for i ← 2 to n do
    if count = 0 then
        current ← A[i]
        counter ← 0
    else
        if current = A[i] then
            counter ← counter + 1
        else
            counter ← counter − 1
done
return current
```

Твърдение: функцията връща най-често срещания елемент от масива A.
За домашно – помислете за инварианта.

Зад. 4. Задачата за най-малкото липсващо естествено число.
Зад. 5. Задачата за k-ти по големина елемент.