DAA Упражнение 3 (Коректност и сложност на алгоритми)

Видовете алгоритми, които ще разглеждаме в курса са или рекурсивни, или итеративни. Коректността на рекурсивен алгоритъм ще правим с помощта на пълна математическа индукция, а коректността на итеративни алгоритми ще доказваме чрез инвариант. Доказателството чрез инвариант е много подобно на доказателството чрез индукция. Разликата е, че И.П. и И.С. са обединени в една стъпка, наречена Поддръжка и има една нова стъпка наречена Терминация. Алгоритмите, които разглеждаме в курса ще пишем на псевдокод, който е със синтаксис много подобен на C/C++ и Python.

зад. 1/ Даден е следният алгоритъм, написан на псевдокод. Какво връща той? Докажете го формално.

```
ALGORITHM1(n: естествено число)          ← := присвояване
1. s ← 1
2. for i ← 1 to n
3.      s ← s*2   // идентираме, вместо да използваме { }
4. return s
```

Решение: Алгоритъмът ALGORITHM1 връща 2^n.

Инвариант: При всяко достигане на ред 2 е изпълнено, че $s = 2^{i-1}$

База: При първото достигане на ред 2, $i = 1$, $s = 1$ от първият ред и проверяваме дали $s = 2^{i-1} \leftrightarrow 1 = 2^{1-1} \leftrightarrow 1 = 1$ ✓

Поддръжка: Нека инвариантът е изпълнен за някое достигане на ред 2, което не е финално. Тогава $s = 2^{i-1}$ ⊛

Изпълнява се тялото на цикъла и $s \leftarrow s*2 \leftrightarrow \underbrace{2^{i-1}}_{\text{от ⊛}}*2 \leftrightarrow 2^i$

При следващо достигане на ред 2 имаме $i' = i+1 \leftrightarrow i = i'-1$ и след заместване на i с $i'-1$ получаваме $s = 2^{i'-1}$, тоест инвариантът е изпълнен спрямо новото $i(i')$.

Терминация: Понеже променливата на цикъла нараства с 1 на всяка итерация и е ограничена отгоре от фиксираната променлива n, то след краен брой итерации ще излезем от цикъла.

При последното достигане на ред 2, $i = n+1$ следователно от инварианта имаме, че $s = 2^{i-1} = 2^{(n+1)-1} = 2^n$. На ред 4 връщаме $s = 2^n$, следователно е изпълнено твърдението ни, че ALGORITHM1 връща 2^n. ∎

Зад. 2/ Даден е следният алгоритъм, написан на псевдокод. Какво връща той? Докажете това формално.

```
ALGORITHM2(A[1,..,n]: масив от цели числа, n ∈ N\{1})
1. sum ← 0
2. for i ← 1 to n
3.     sum ← sum + A[i]
4. return sum
```

$A[1,\dots,n]$ – n-мерен масив
$A[1,\dots,n,1,\dots,m]$ – масив с размер m на n
индексираме масивите от 1 за удобство

Решение: Алгоритъмът ALGORITHM2 връща $\sum_{j=1}^{n} A[j]$

Инвариант: При всяко достигане на ред 2 е изпълнено, че $sum = \sum_{j=1}^{i-1} A[j]$

База: При първото достигане на ред 2, $i = 1$ и $sum = 0$ от инициализацията на 1 ред. Питаме се дали $sum = \sum_{j=1}^{0} A[j] = 0$. Да, понеже в сумата не сме сумирали нищо и нулата е неутрален елемент спрямо събирането.

Поддръжка: Нека инвариантът е изпълнен за някое достигане на ред 2, което не е финално. Тогава $sum = \sum_{j=1}^{i-1} A[j]$ (*)

Изпълнява се тялото на цикъла и имаме $sum \leftarrow sum + A[i]$, което е същото като $sum \leftarrow \underbrace{\sum_{j=1}^{i-1} A[j]}_{\text{от } (*)} + A[i] \leftrightarrow \sum_{j=1}^{i} A[j]$. Следователно при следващо достигане на ред 2 с $i' = i+1 \leftrightarrow i = i'-1$ ще имаме, че sum съдържа $\sum_{j=1}^{i'-1} A[j]$ и инвариантът е верен спрямо новото i (i').

Терминация: Понеже променливата на цикъла расте и е ограничена отгоре от фиксирано n, то след краен брой итерации ще излезем от цикъла. При последното достигане на ред 2 имаме, че $i = n+1$ и от инвариантът имаме, че $sum = \sum_{j=1}^{i-1} A[j] = \sum_{j=1}^{n} A[j]$ и на ред 4 връщаме sum, следователно твърдението е доказано. ∎

Забележка: Доказателствата с инвариант са средство с което доказваме коректността на итеративен цикъл, а не на целият алгоритъм! Доказателствата чрез пълна математическа индукция доказват коректността на целия алгоритъм, но ако в него има някакви цикли, то трябва първо да докажем коректността им чрез инвариант.

зад. 3/ Даден е следният рекурсивен алгоритъм, написан на псевдокод. Какво връща той? Докажете го формално.

```
RECALG(a ∈ R⁺, n ∈ N)
1. if n = 0 then
2.     return 1
3. if n is even then // n ≡ 0 (mod 2)
4.     return RECALG(a*a, n/2)
5. return a * RECALG(a, n-1)
```

Решение: Алгоритъмът RECALG връща a^n. Това ще докажем чрез пълна математическа индукция по n.

База: $n = 0$ От 1 и 2 ред имаме, че ако $n = 0$, то алгоритъмът връща 1, което е същото като a^0. ✓

И.П.: Нека $\forall m \le n$ е изпълнено, че RECALG(a, m) връща a^m

И.С.: Доказваме твърдението за $n+1 > 0$ (>0 защото да не гледаме отново 1,2 ред т.е базата)

I случай) Ако $n+1$ е четно, то ~~[illegible]~~ условието на 1 ред е лъжа, понеже $n+1 > 0$ и не се изпълнява тялото на if-a (ред 2). Понеже $n+1$ е четно, то ред 3 е истина и ще се изпълни ред 4 където се връща RECALG$(a^2, \frac{n+1}{2})$. От И.П. имаме, че $\forall m \le n$ RECALG$(a, m) = a^m$ и $\frac{n+1}{2} \le n$, понеже $n > 1$, тогава се връща $(a^2)^{\frac{n+1}{2}} = a^{n+1}$ ✓

II случай) Ако $n+1$ е нечетно, то условието на ред 1 е лъжа, понеже $n+1 > 0$ и не се изпълнява ред 2. Понеже $n+1$ е нечетно, то ред 3 е лъжа и ред 4 не се изпълнява. Ред 5 се изпълнява и се връща $a *$ RECALG$(a, n+1-1)$, но от $\boxed{\text{И.П.}}$ имаме, че $\forall m \le n$ RECALG$(a, m) = a^m$, тогава на ред 5 се връща $a * a^n = a^{n+1}$ ✓

I и II случай са изчерпателни, следователно твърдението е доказано. ■

зад.4/ Даден е следният алгоритъм написан на псевдокод. Какво връща той? Докажете това формално и прецизно.

```
KADANE(A[1,...,n]: масив от цели числа)
1. curr ← A[1]
2. best ← A[1]
3. for i ← 2 to n          // i ← n downto 2, ако искахме наобратно
4.     if A[i] + curr > A[i] then
5.         curr ← curr + A[i]
6.     else
7.         curr ← A[i]
8.     if curr > best
9.         best ← curr
10. return best
```

Решение: Алгоритъмът KADANE връща най-голяма (не най-голямата) сума на непразен подмасив (последователни елементи) на A[1...n].

Инвариант: При всяко достигане на ред 3 е изпълнено, че curr съдържа най-голяма сума на непразен подмасив на A[1,...,i-1], завършващ в индекс i-1 и best съдържа най-голяма сума на непразен подмасив на A[1,...,i-1] (не трябва непременно да завършва в индекс i-1).

Докажете База, Поддръжка и Терминация за домашно, като разгледате всички възможни случаи! Целта на този пример е да подчертаем, че инвариантът трябва да включва информация за нещата, които ще ни помогнат за доказателството и да сме прецизни с изказа.