ДАА Семинар 4

Коректност на итеративни алгоритми

За да докажем коректността на даден алгоритъм, ни трябва прецизно описание на проблема, който съответният алгоритъм е предназначен да решава.

Най-лесният начин да докажем, че алгоритъмът не е коректен е чрез контрапример - възможно най-простият конкретен вход, за който алгоритъмът не генерира очакваният изход.

Обща схема

На всеки „нетривиален" цикъл съпоставяме инвариант
Инвариант - предикат P(k), където k е броят достигания на реда с условието на цикъла, тоест $k \in \{1, \dots, m\}$, ако съответният ред се достига m пъти, или $k \in \{a, \dots, b+1\}$, ако итераторът на цикъла заема стойностите от $\{a, \dots, b+1\}$.

Задачата е да докажем верността на P(k) по индукция върху краен домейн:

Три етапа:

1) База - разглеждаме първото достигане на съответния ред и доказваме P(1)
2) Поддръжка - допускаме P(t) и доказваме P(t+1); t+1 е в домейна
3) Терминация - разглеждане последното достигане на съответния ред и използване вече доказаното твърдение P(k)

зад ① Да се разгледа следния алгоритъм. Какво връща той? а) Да се докаже б) не / формално а)

```
Alg1 (int n)
{
1. int s = 1;
2. for(int i = 0; i<n; i++)
3.       s=s*2;
4. return s;
}
```

Решение:

a) Alg1 връща 2^n

б) Ще докажем, че Alg1 връща 2^n при извикване: Alg1(n)

Инвариант на цикъла: При k-тото достигане на ред 2 s има стойност 2^{k-1}

Доказателство:

1) База: При първото достигане на ред 2 (k=1) i=0 и $s=1=2^0=2^{k-1}$ ✓

2) Поддръжка: Допускане, че някое k-то достигане на ред 2, което не е последното е изпълнено: $s=2^{k-1}$.

Тогава i ще има стойност k-1. В тялото на цикъла s заема нова стойност - 2^k. На следващото $^{k+1\text{-во}}$ достигане на ред 2 s има стойност 2^{k-1}, изразено в стойността на новото k.

3) Терминация: Последното достигане на ред 2 е (n+1)-вото и тогава s ще има стойност 2^n. Цикълът не се изпълнява повече пъти. Непосредствено след това алгоритъмът връща $s=2^n$ като резултат ▯.

Зад ② Да се разгледа следния алгоритъм.

```
Alg2 (int a[n])
{
1. int s = 0;
2. for(int i=0; i<n; i++)
3.     s = s + a[i];
4. return s;
}
```

а) Какво връща той?

б) Да се докаже а)

Решение:

а) Alg2 връща сумата на елементите на масива a при извикване Alg2(a,n)

б) Инварианта на цикъла: При k-тото достигане на ред 2 s има стойността - $\sum_{j=0}^{k-2} a[j]$ - сумата на елементите в масива a[0...k-1]

Доказателство:

1) База: Разглеждане първото достигане на ред 2. От една страна в текущия момент s=0, от друга - масивът a[0...k-2] = a[0...-1] е празният масив, а сумата на елементите на празния масив е 0 - неутралният елемент на събирането ✓

2) Поддръжка. Разглеждане някое k-то достигане на ред 2, което не е последно. Допускаме, че в този момент s съдържа сумата на елементите в подмасива a[0...k-2]. В тялото на цикъла s приема нова стойност - сумата на елементите в подмасива a[0...k-2, i=k-1]. Тази стойност се запазва до следващото - k'=k+1-во достигане на ред 2. Изразено в k' тогава s има стойността - сумата на елементите в подмасива a[0...k'-2]

3) Терминация: При последното достигане на ред 2 $k=n+1$. От верността на инварианта следва, че в този момент S има за стойност сумата на елементите в $a[0...k-2] = a[0...(n+1)-2] = a[0...n-1]$, което е точно сумата на елементите във входния масив. Цикълът не се изпълнява повече, непосредствено след това #g2 връща като резултат S.

зад ③ Да се разгледа следната функция, която връща n^3

```
int foo (int n)
{
1.  int i = 0;
2.  int x = 6, y = 1, z = 0;
3.  for (i; i<n ; i++)
4.  {
5.    z += y;
6.    y += x;
7.    x += 6;
8.  }
9.  return z;
}
```

Да се докаже следния инвариант на цикъла:

При всяко достигане на ред 3. променливата
z съдържа i^3, y съдържа $3i^2+3i+1$, x съдържа $(i+1).6$.

Доказателство:

1) База: Разглеждане първото достигане на ред 3. В този момент $i=0 \wedge$
$x = 6 = (0+1).6 \wedge y = 1 = 3.0^2+3.0+1 \wedge z = 0 = 0^3$ ✓

2) Поддръжка: Допускане, че инвариантът е изпълнен за някое достигане на ред 3, което не е последното. В този момент е изпълнено:
$z = i^3 \wedge y = 3i^2+3i+1 \wedge x = (i+1).6$ за текущото i. Следва на влизане в цикъла (щом достигането не е последно (от предположението)). Изпълняват се редове 5, 6, 7.
След които новите стойности на променливите са съответно:

$z = i^3 + 3i^2+3i+1 = (i+1)^3$

$y = 3i^2+3i+1 + (i+1)6 = 3i^2+3i+1+6i+6 = 3i^2+6i+3+3i+3+1 = 3(i^2+2i+1)+3(i+1) = 3(i+1)^2+3(i+1)+1$ (+1 ✓)

$x = (i+1).6+6 = (i+2).6$

Изпълнението отива на ред 3, където i се инкрементира. Спрямо стойността на новото $i'=i+1$ е в сила:

$z = i'^3$

$y = 3i'^2+3i'+1$

$x = (i'+1).6$

С което доказахме инварианта.

зад ④ Да се разгледа следния алгоритъм с линейна сложност, който намира максимална сума на елементите на подмасив:

```
Kadane(A[1...n])
1. sum ← 0
2. max ← 0
3. for i ← 1 to n
4.     sum ← sum + A[i]
5.     if sum > max
6.         max ← sum
7.     if sum < 0
8.         sum ← 0
9. return max
```

// Алгоритъмът избира да „влачи" сбора на елементите на текущия разглеждан подмасив, само ако е положителен - тогава ще допринесе, евентуално, за по-голям сбор на друг по-голям масив - разширение на текущия. Ако сборът е отрицателен, той би „сволял" от „теглото" на следващите елементи, тоест, няма смисъл да се разширява текущия подмасив, който има отрицателен сбор.

// Пример #7

$A = \{-2; -3; \boxed{4; -1; -2; 1,5}; -3\}$

Ако изберем да „носим" стойността -2 (тоест да се опитваме да разширяваме текущия под масив $\{-2\}$), то тя само ще намали стойността на следващите суми. Същото важи и за стойността -3.

Избиране да „носим" стойността 4, тъй като тя ще допринесе към търсената сума. // На ред 4: sum ← 4 + (-1) = 3 - тази сума също е положителна и избиране да я „носим" ... 3 + (-2) = 1 > 0 - също допринася.

За конкретния пример максималната сума на елементите на подмасив (може да има няколко такива суми) е 7, а съответният подмасив е $\{4, -1, -2, 1, 5\}$.

- Да се докаже коректността с инвариант на цикъла

Инвариант: При всяко достигане на ред 3 променливата sum съдържа максималната сума на подмасив в $A[i..i-1]$, завършващ с $A[i-1] =: MAX_{1,i-1}^{i-1}$ (този подмасив може и да е празният), а променливата max - максималната сума на подмасив в $A[1..i-1] =: MAX_{1,i-1}$

Доказателство на инварианта:

1) База: Разглеждаме първото достигане на ред 3. В текущия момент

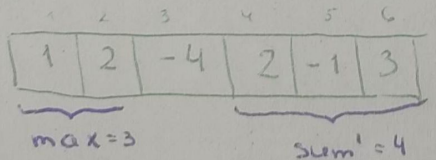
е в сила: $i = 1 \wedge sum = 0 \wedge max = 0 \wedge A[1..i-1] = A[1..0] = []$.
Единственият подмасив на $A[1..0] = []$ е самият празен масив. Като единствен негова та сума е максимална. Сумата на елементите на празния масив е 0 - неутралният елемент на операцията максимум. ✓

2) Поддръжка: Допускаме, че твърдението (инварианта) е изпълнено за някое достигане на ред 3, което не е последното. Щом достигането на ред 3 не е последното, то следва навлизане в тялото на цикъла. На ред 4. променливата sum заема нова стойност $sum' = sum + A[i]$. Разглеждаме ред 5. Има две възможности:

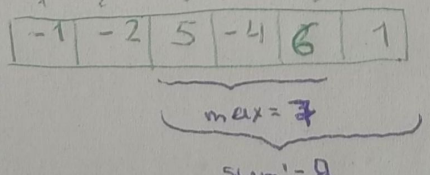
1) $sum' > max$ е истина, което означава, че съществува подмасив, завършващ в $A[i]$, със сума, по-голяма от стойността на max, която, спрямо предположението, съдържа $MAX_{1,i-1}$ (сумата на подмасив в $A[1..i-1]$ с максимален сбор). Тогава $sum' = MAX_{1,i}$. в този момент.

// Последното ($sum' = MAX_{1,i}$) е в сила, тъй като (7) чрез max ✓ е сумата на някой подмасив с максимална сума, който не завършва в $A[i-1]$ и тогава няма как да му „долепим" елемента $A[i]$, тъй като това не би бил непразният подмасив и означава, че sum' съдържа максималната сума на подмасив, ~~ж~~ е $A[1..i]$ завършващ в $A[i]$ и е новият max, (6) чрез max в текущия момент е сумата на някой подмасив с максимална сума, който завършва в $A[i-1]$ и може да го разширим с елемента $A[i]$ (който в този случай е положителен).

// Пример за (7): В текущия момент $i = 6$ и масивът $A[1..6]$ изглежда така



// Пример за (6): В текущия момент $i=6$ и масивът $A[1..6]$ е:



Изпълнява се ред 6 и $max = MAX_{1,i}$ (Δ)

2) $sum' > max$ е лъжа, следователно $sum' \leq max$ е в сила и тогава, спрямо допускането $max = MAX_{1,i}$ (елементът $A[i]$ не влияе на текущия максимум.) В този случай ред 6 не се изпълнява и max не променя стойността си. (*)

=> и в двата случая (от (Δ) и (*)) => $max = MAX_{1,i}$

Разглеждане ред 7. Има две възможности:

1) $sum' < 0$ е истина

Това означава, че всеки непразен подмасив на $A[1..i]$ с най-десен елемент $A[i]$ има отрицателен сбор на елементите. Тогава максималният сбор на подмасив, завършващ с $A[i]$ е 0 и съответства на празния масив. В този случай се изпълнява ред 8 и стойността на sum' става точно 0. (Δ)

2) $sum' \geq 0$ е истина

От това и от допускането, че $sum = MAX_{1,i-1}^{i-1}$ (максималния сбор на подмасив на $A[1..i-1]$, завършващ в $A[i-1]$), следва, че sum' съдържа $MAX_{1,i}^{i}$ // Ако допуснем противното: че съществува друг подмасив, завършващ на $A[i]$ със сума $sum2 > sum'$ ще получим противоречие с допускането (*)

=> от (Δ) и (*) следва, че $sum' = MAX_{1,i}^{i}$

При следващо достигане на ред 3 i се увеличава с единица: $i' = i+1$ и спрямо новото i е в сила, че в текущия момент е изпълнено:

$$max = MAX_{1,i'-1} \wedge sum' = MAX_{1,i'-1}^{i'-1}$$

3) Терминация: Разглеждане последното достигане на ред 3. В текущия момент i има стойност $n+1$ и спрямо доказания инвариант:

max съдържа $MAX_{1,n}$ - максимална сума на подмасив на $A[1...(n+1)-1] = A[1..n]$ Тъй като условието за навлизане в цикъла е лъжа, изпълнението отива на ред 9, където max бива връщана като резултат. ▫

Решения на задачите за упражнения:

① Да се изчисли стойността на count след изпълнение на следния алгоритъм

```
AlgX(n=2^k, k∈ℕ)
1. count ← 0
2. i ← 1
3. while i≤n
4.     for j ← 1 to i
5.         count ← count+1
6.     i ← 2i
7. return count
```

Решение:

Разглеждаме стойностите, заемани от итератора на while цикъла:

$i = 1, 2, 4, \ldots, n$

Може да запишем горното по следния начин:

$i = 2^0, 2^1, 2^2, \ldots, 2^k = n$

По-удобно е да работим с итератори, които се увеличават с единица, затова полагаме $i = 2^r \Rightarrow r = \log i$, където $r = 0, 1, 2, \ldots, k$

Спрямо новия итератор получаваме следната вложена сума:

$$\sum_{r=0}^{k}\sum_{j=1}^{i} 1 = \sum_{r=0}^{k} i = \sum_{r=0}^{k} 2^r \overset{(\Delta)}{=} \frac{2^{k+1}-1}{2-1} = 2^{k+1} - 1 = 2^{\log n + 1} - 1 = 2 \cdot 2^{\log n} - 1 = 2n - 1 \quad \blacksquare$$

// (Δ) Използване формулата за сумата на първите n члена на геометрична прогресия: $S_n = a_1 + a_2 + \ldots + a_n = a_1 \frac{q^n - 1}{q-1}$ // в случая броим от 0 и търсим първите k+1 члена

② AlgY(n=2^{2^k}, k∈ℕ)

```
1. count ← 0
2. for i ← 1 to n
3.     j ← 2
4.     while j≤n
5.         j ← j^2
6.         count ← count+1
7. return count
```

Решение:

Разглеждаме стойностите, заемани от итератора j:

$j = 2, 4, 16, 256, \ldots$, тоест $j = 2^1, 2^2, 2^{2^2} = (2^{2^2})^2, \ldots$, тоест $j = 2^{2^0}, 2^{2^1}, 2^{2^2}, 2^{2^3}, \ldots, 2^{2^k}$

Полагане r: $j = 2^{2^r}$, тоест $r = \log\log j$. Получаване сумата

$$\sum_{i=1}^{n}\sum_{r=0}^{k} 1 = \sum_{i=1}^{n}(k+1) = \sum_{i=1}^{n}(\log\log n + 1) = (\log\log n + 1)\sum_{i=1}^{n} 1 = n\log\log n + n$$