Дизайн и Анализ на Алгоритми

Примерни решения на малко контролно № 1

40 т. **Задача 1.** Предложете колкото се може по-бърз алгоритъм (в асимптотичния смисъл), чийто вход е произволен **целочислен** масив $A[1,\ldots,n]$ и изход максимално произведение на подмасив на $A[1,\ldots,n]$. Напишете алгоритъма на псевдокод, формулирайте инвариант, дайте кратка аргументация за коректност и направете анализ на сложността по време и по памет.

20 т. Модифицирайте вашия алгоритъм да връща индексите на подмасива, освен максимално произведение. Дайте кратка аргументация (не е нужен инвариант).

Решение. Следният псевдокод реализира исканото:

Algorithm 1 Max Product Subarray

```
1:  currMaxProduct ← A[1]
2:  currMinProduct ← A[1]
3:  globalMaxProduct ← A[1]
4:  for i ← 2 to n do
5:      if A[i] < 0 then
6:          SWAP(currMaxProduct, currMinProduct)
7:      end if
8:      currMaxProduct ← max(A[i], currMaxProduct ∗A[i])
9:      currMinProduct ← min(A[i], currMinProduct ∗A[i])
10:     globalMaxProduct ← max(globalMaxProduct, currMaxProduct)
11: end for
12: return globalMaxProduct
```

Инвариант: При всяко достигане на ред 4 следните твърдения са в сила:

- Променливата `currMaxProduct` пази **максимално** по стойност произведение на подмасив, *завършващо в индекс* $i-1$.
- Променливата `currMinProduct` пази **минимално** по стойност произведение на подмасив, *завършващо в индекс* $i-1$.
- Променливата `globalMaxProduct` пази **максимално** по стойност произведение в подмасива $A[1,\ldots,i-1]$

Кратка обосновка: Предложеното решение взаимства идеята на алгоритъма на Kadane, разглеждан на упражнения. Ключовото наблюдение е, че произведението на четен брой отрицателни числа е положително число. При това съобразяваме, че всяко произведение, в което участва нула, е нула. Очевидно сложността на предложения алгоритъм е $\Theta(n)$: тялото на for-цикъла на ред 4 се състои от елементарни операции (присвоявания и сравнявания) с времева сложност $\Theta(1)$, а общият брой итерации е точно $n-1$. Дефинират се три променливи, т.е. сложността по памет е $\Theta(1)$.

Algorithm 2 Max Product Subarray With Indices

```
currMaxProduct ← A[1]
currMinProduct ← A[1]
globalMaxProduct ← A[1]
leftCurrMaxIndex← 1
rightCurrMaxIndex ← 1
leftCurrMinIndex← 1
rightCurrMinIndex ← 1
leftGlobalMaxIndex← 1
rightGlobalMaxIndex ← 1
for i ← 2 to n do
    if A[i] < 0 then
        SWAP(maxProductAtIndex, minProductAtIndex)
        SWAP(leftCurrMaxIndex, leftCurrMinIndex)
        SWAP(rightCurrMaxIndex, rightCurrMinIndex)
    end if
    if A[i] > currMaxProduct ∗A[i] then
        currMaxProduct ← A[i]
        leftCurrMaxIndex ← i
        rightCurrMaxIndex ← i
    else
        maxProductAtIndex ← currMaxProduct ∗A[i]
        rightCurrMaxIndex ← i
    end if
    if A[i] < currMinProduct ∗A[i] then
        currMinProduct ← A[i]
        leftCurrMinIndex ← i
        rightCurrMinIndex ← i
    else
        currMinProduct ← currMinProduct ∗A[i]
        rightCurrMinIndex ← i
    end if
    if globalMax < currMaxProduct then
        globalMax ← currMaxProduct
        leftGlobalMaxIndex ← leftCurrMaxIndex
        rightGlobalMaxIndex ← rightCurrMaxIndex
    end if
end for
return globalMax, leftGlobalMaxIndex, rightGlobalMaxIndex
```

Кратка аргументация: За всяка променлива пазим ляв и десен индекс, които отбелязват съответно началото и края на подмасива.

40 т. **Задача 2.** Предложете колкото се може по-бърз алгоритъм (в асимптотичния смисъл), чийто вход е произволен **сортиран** масив $A[1, \ldots, n]$ с **уникални** елементи и число $k \in \mathbb{Z}$, който връща "Да", ако съществува индекс $i \in \{1, \ldots, n\}$ за който е изпълнено $i = A[i] - k$, и "Не" в противен случай. Напишете алгоритъма на псевдокод, дайте кратка аргументация за коректност и направете анализ на сложността по време и по памет.

Примери: При вход $A = [-11, -4, 2, 4, 6]$ и $k = -1$, алгоритъмът връща "Да" (понеже $3 = A[3] - k$). При вход $A = [2, 3, 4, 5, 6, 7]$ и $k = 0$, алгоритъмът връща "Не".

Решение: Задачата се решава чрез непосредствена модификация на алгоритъма за двоично търсене в сортиран масив:

Algorithm 3 Binary Search Adaptation

```
left ← 1
right ← n
while left <= right do
    mid ← ⌊(left+right)/2⌋
    if mid = A[mid] − k then
        return "Да"
    else if mid < A[mid] − k then
        right = mid − 1
    else
        left = mid + 1
    end if
end while
return "Не"
```

Кратка аргументация: Сложността по време е $\Theta(\lg n)$, сложността по памет е $\Theta(1)$.