Недетерминирани Алгоритми

Деф. Ще казваме, че недетерминиран алгоритъм връща TRUE, ако **поне един** клон от изпълнението му връща TRUE. В противен случай ще връщаме FALSE.

Зад. 1 (COMPOSITES)

Даден е масив $A[1 .. n] \in \{0, 1\}^n$ със свойството, че $n \geq 2 \Rightarrow A[1] = 1$. Семантиката на този масив е число в двоична бройна ситема. Да се провери дали числото е съставно.

***a*)** чрез детерминиран алгоритъм

***b*)** чрез недетерминиран алгоритъм

Пример:

$\begin{cases} \text{Вход : } A[1 ..6] = [1, 0, 1, 1, 1, 0] \text{ // т.е. } 101\,110_{(2)} = 46_{(10)} \\ \text{Изход : да} \end{cases}$

***a*)**

```
1. isCompositeDet(A[1 .. n]) : // A ∈ {0, 1}^n, (n ≥ 2 → A[1] = 1)
2.       num ← binToDec(A) // num ∈ {0, 1, …, 2^n − 1}
3.       for q ← 2 to ⌊√num⌋
4.                 if num ≡ 0 (mod q) then
5.                           return TRUE
6.       return FALSE
```

Коректност *Това ни е добре познато как се прави - чрез инвариант*

Сложност Големината на входа ни е n.

· Ред 2 ни е с линейна (или това което ни интересува - полиномиална) сложност.

· Цикъла се върти най-много $\theta\left(\sqrt{2^n}\right) = \theta\left(\left(\sqrt{2}\right)^n\right)$, а изпълнението на тялото му отнема $\theta(1)$ работа.

Тоест isCompositeDet($A[1 .. n]$) = $\theta(n) + O\left(\left(\sqrt{2}\right)^n\right)$. Интересува ни най-лошия случай - сложността е експоненциална.

***b*)**

```
1. isCompositeNonDet(A[1 .. n]) : // A ∈ {0, 1}^n, (n ≥ 2 → A[1] = 1)
2.       num ← binToDec(A) // num ∈ {0, 1, …, 2^n − 1}
3.       q ← choose({2, 3, …, ⌊√num⌋})
4.       if num ≡ 0 (mod q) then
5.                 return TRUE
6.       return FALSE
```

Факт Нека $A = \{a_1, \ldots, a_m\}$ е множество. Тогава choose(A) е с логаритмична сложност спрямо m. Защо? (нетривиално е - трябват познания за Машини на Тюринг)

Коректност Трябва да покажем: $\begin{cases} \text{поне едно изпълнение на функцията връща TRUE} & \text{, num – съставно} \\ \text{всички изпълнения на функцията връщат FALSE} & \text{, num – просто} \end{cases}$.

· Функията binToDec я приемаме за коректна.

· Нека num = $a.b$, като $a, b > 1$. Нека без ограничение на общността $a \leq b$. Тогава е ясно, че $a \leq \lfloor \sqrt{\text{num}} \rfloor$. На ред 3 присвояваме на q стойност от множеството $\{2, \ldots, \lfloor \sqrt{\text{num}} \rfloor\}$. Нека разгледаме клона на изпълнение в който choose$(\{2, 3, \ldots, \lfloor \sqrt{\text{num}} \rfloor\})$ ни връща a. Тогава е ясно, че алгоритъма ще върне TRUE. Тоест поне един клон върна TRUE ⇒ нед. алг. връща TRUE.

· Нека num е просто. Тогава num няма делители. Тогава за всеки избор choose$\left(\left\{2, 3, \ldots, \left\lfloor\sqrt{\text{num}}\right\rfloor\right\}\right)$ функцията ще върне FALSE.

Тоест нед. алг. връща FALSE.

Сложност Големината на входа ни е n.

· Ред 2 ни е с линейна (или това което ни интересува - полиномиална) сложност.

· Ред 3 ни е полиномиална сложност спрямо $\sqrt{n}$. Тоест е и полиномиална спрямо големината на входа ни n.

· Редове 4-6 са ни константа работа

Тоест isCompositeNonDet($A[1 .. n]$) = $\theta(n) + \theta\left(\sqrt{n}\right) + \theta(1) = \theta(n)$. Интересува ни най-лошия случай - сложността е полиномиална.

Нека да разгледаме обратната задача на COMPOSITES - PRIMES.

Зад. 2 (PRIMES)

Даден е масив $A[1 .. n] \in \{0, 1\}^n$ със свойството, че $n \geq 2 \Rightarrow A[1] = 1$. Семантиката на този масив е число в двоична бройна ситема. Да се провери дали числото е просто.

***a*)** чрез детерминиран алгоритъм

***b*)** чрез недетерминиран алгоритъм

Пример:

$\begin{cases} \text{Вход : } A[1 ..6] = [1, 0, 1, 1, 1, 0] \text{ // т.е. } 101\,110_{(2)} = 46_{(10)} \\ \text{Изход : не} \end{cases}$

***a*)**

```
1. isPrimeDetNaive(A[1 .. n]) : // A ∈ {0, 1}^n, (n ≥ 2 → A[1] = 1)
2.       return not isCompositeDet(A)
```

Коректност Тъй като е дуалната задача по тривиални причини е коректна

Сложност При най-лоши входни данни - експоненциална

Нека да разгледаме, че при нед. алг. **НЕ** може да правим така (99.9 % от случаите).

***b*)**

```
1. isPrimeNonDetNaive(A[1 .. n]) : // A ∈ {0, 1}^n, (n ≥ 2 → A[1] = 1)
2.       return not isCompositeNonDet(A)
```

Коректност За да е коректен нед. алг. трябва да покажем: $\begin{cases} \text{поне едно изпълнение на функцията връща TRUE} & \text{, num – просто} \\ \text{всички изпълнения на функцията връщат FALSE} & \text{, num – съставно} \end{cases}$.

Ще покажем контрапример, откъдето ще следва, че нед. алг. **не** е коректен.

· Нека num = 16 (т.е е съставно). Ще покажем, че има клон на изпълнение, който ни връща TRUE. Нека choose({2, 3, 4}) ни е избрало 3. Тогава isCompositeDet([1, 0, 0, 0, 0]) ни връща FALSE. Съответно функцията isPrimeNonDetNaive ще ни върне TRUE. Тоест не всички изпълнения връщат FALSE за всички съставни числа ⇒ нед. алг. е некоректен.

Забележка Изобщо не е тривиално да се състави полиномиален недетерминиран алгоритъм, който да решава задачата PRIMES. Като цяло в общия случай ако една задача за разпознаване има нед. алг. с полиномиална сложност, то не е гарантирано, че обратната ѝ ще има нед. алг. с полиномиална сложност.

Заключение:

· PRIMES $\overset{?}{\in}$ NP

· $\overline{\text{PRIMES}}$ = COMPOSITES ∈ NP ⇒ PRIMES ∈ co − NP

Факт Доказано (изобщо нетривиално) е, че:

1) PRIMES ∈ NP **2)** PRIMES ∈ P **3)** PRIMES ∈ co − P