

Въведение в алгоритмите.

Сложност на алгоритъм



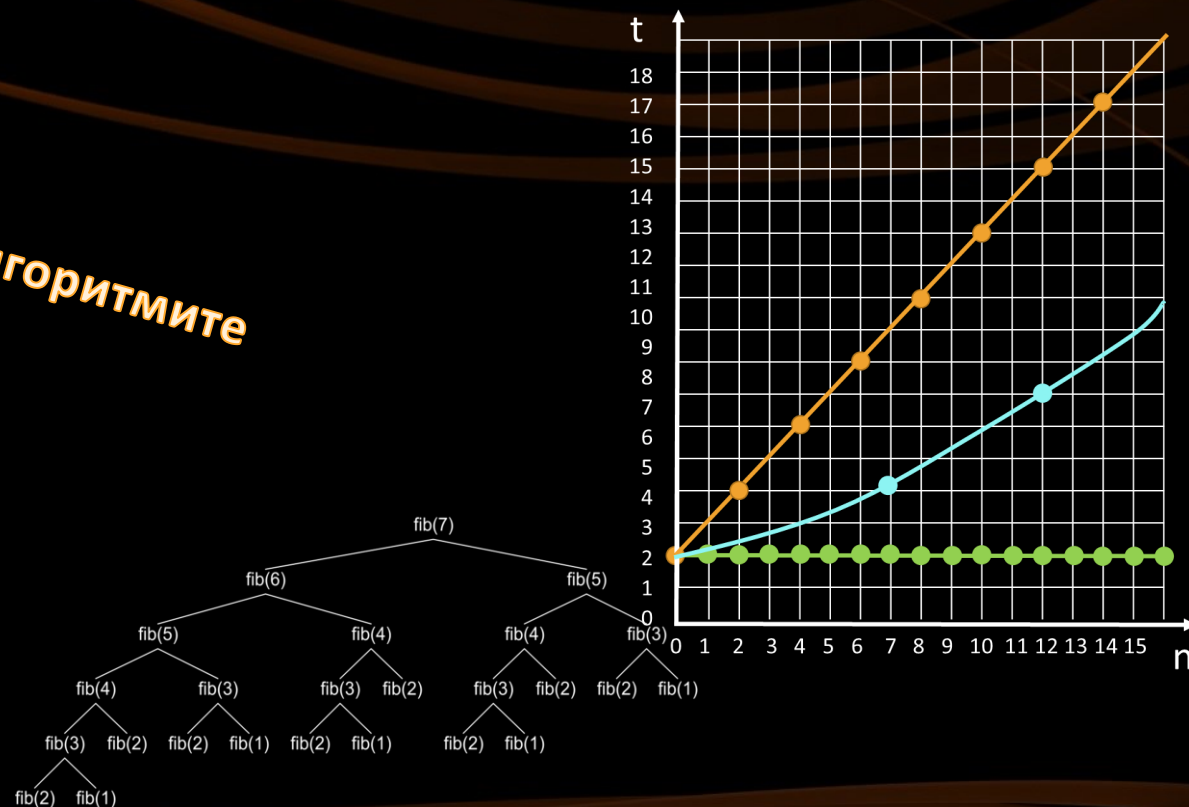
Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>



Увод в алгоритмите



Съдържание

1. Анализ на алгоритъм. Брой операции в алгоритъм
 - Ресурси : CPU, RAM, шина, външна памет, други
2. Най-добър, среден и най-лош случай
3. Опростяване и намаляване на броя на стъпките
4. Задачи за изчисляване на сложност



Анализ на алгоритъм

- Определение за алгоритъм (неформално)
 - краен брой, еднозначно определени стъпки (команди), водещи до решаването на даден проблем
- Видове алгоритми – линейни, разклонени
- По-важни свойства на алгоритмите:
 - дискретност
 - определеност
 - крайност
 - масовост
 - **СЛОЖНОСТ**



Анализ на алгоритъм(2)

- Защо трябва да анализираме алгоритмите?
 - Предсказване на необходимите **ресурси** за алгритъма
 - Изчислително време (работа на **CPU**)
 - Необходимо количество оперативна памет(**RAM**)
 - Ползване на **честотната лента (шина)** за комуникация
 - Операции с **твърдия диск**
 - Употреба на **всякакви времеемки и енергоемки ресурси**



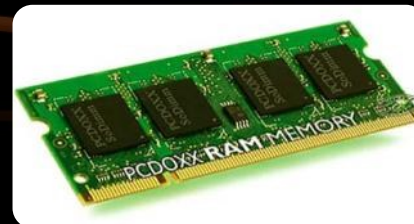
Анализ на алгоритъм(3)

- Очакваното **време** за изпълнение на алгоритъма е:
 - Общият брой изпълнени **елементарни операции**
(машинно-зависими стъпки)
 - Също позната като **сложност на алгоритъма**
 - Сравняване на алгоритмите, като се **изключват детайлите като** език или хардуер



Сложност на алгоритъм

- Какво измерва?
 - CPU време
 - Консумация на памет
 - Брой стъпки
 - Брой частични операции
 - Брой дискови операции
 - Брой мрежови пакети
 - Асимптотична сложност



Задача: Намерете сбора от стъпките

- Изчислете максималния брой стъпки за намиране на сбора от нечетните елементи в масив

```
int GetSumEven(int[] array)
{
    int sum = 0;
    for (int i = 0; i < array.Length; i++)
        if (array[i] % 2 == 0) sum += array[i];
    return sum;
}
```

Решение:

$$T(n) = 9n + 3$$

Изчисляването на максималния брой стъпки се нарича анализ **Най-лош случай**

- Предполагаме, че **една стъпка** е една инструкция на CPU:
 - Подредби, търсения на елемент в масив, сравнения, аритметични операции

Време за изпълнение и екстремални случаи

■ Най-лош случай

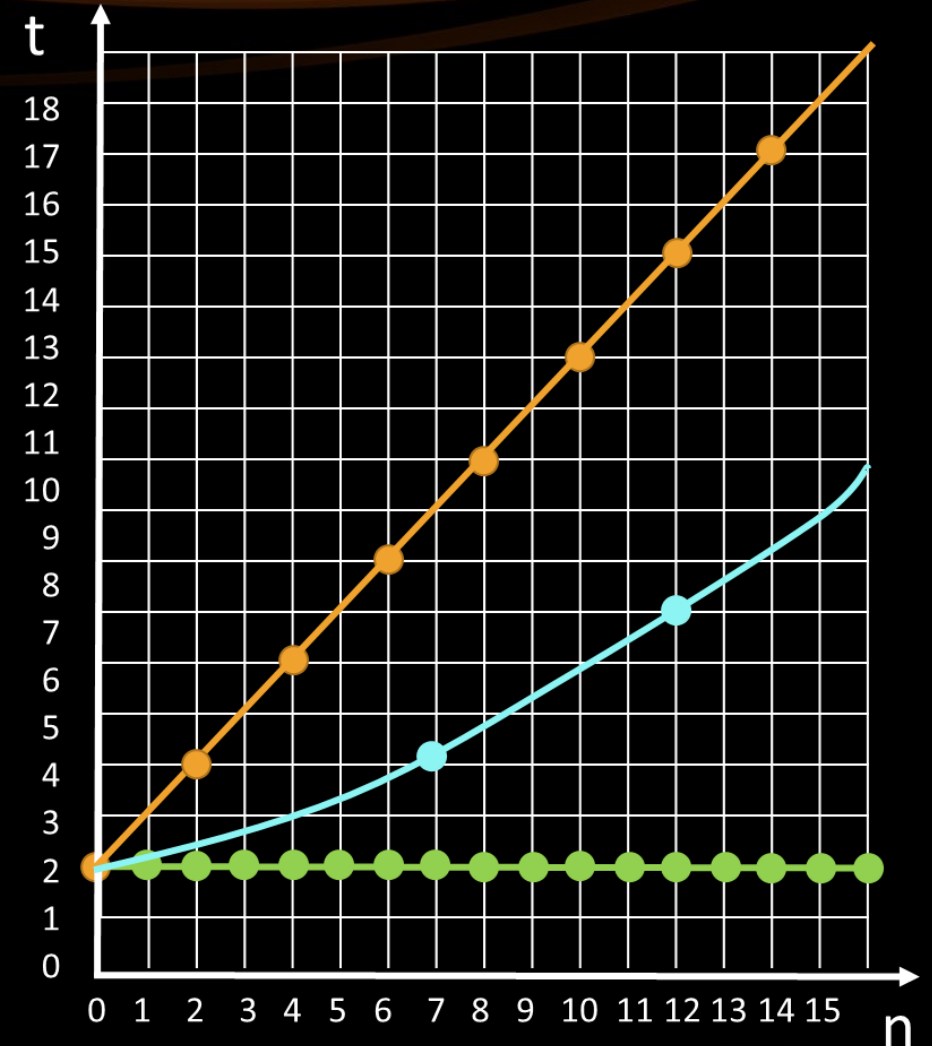
- Горна граница на времето за изпълнение

■ Средно аритметичен случай

- Средно време за изпълнение

■ Най-добър случай

- Долна граница на времето за изпълнение (оптимален случай)



Задача: Изчисляване броя на стъпки

- Изчислете максималния брой стъпки за намиране на съществуващ в масива елемент

```
int Contains(int[] array, int element)
{
    for (int i = 0; i < array.Length; i++)
        if (array[i] == element) return true;
    return false;
}
```

Решение:
 $T(n) = 4n + 4$

- Предполагаме, че **една стъпка** е една инструкция на CPU, като:
 - подредба, търсене в масив, сравнения, аритметични операции

Опростяване и намаляване на броя стъпки

- Някои изрази нарастват много по-бързо от други

```
int Contains(int[] array, int element)
{
    for (int i = 0; i < array.Length; i++)
        if (array[i] == element) return true;
    return false;
}
```

$n = 1000$
стъпки: $4000 + 4$

Съответна част: n

- По-високия степенен показател **доминира** по-малкия степенен показател – $n > 2$, $n^2 > n$, $n^3 > n^2$
- Константните множители може да бъдат **пропуснати** – $12n \rightarrow n$, $2n^2 \rightarrow n^2$

Задача: Брой стъпки в задачата Фибоначи

- Изчислете стъпките за намиране на $n^{\text{ти}}$ член на редицата от числа на Фибоначи рекурсивно

```
int Fibonacci(int n)
{
    if (n == 0) return 1;
    else if (n == 1) return 1;
    else return Fibonacci(n-1) + Fibonacci(n-2);
}
```

Решение:

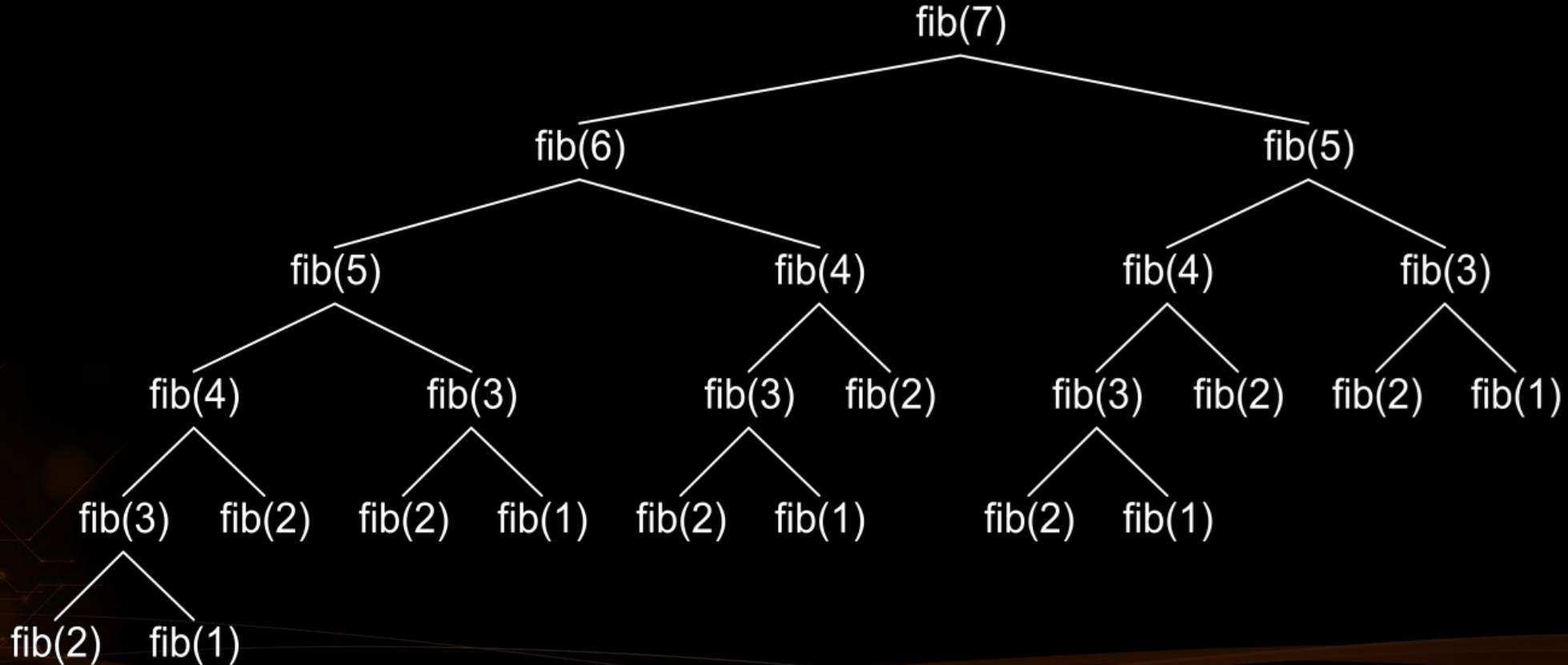
$$T(n) = 3 + T(n - 1) + T(n - 2)$$

- но $F_n = F_{n-1} + F_{n-2}$, което е $\leq T(n)$
- обаче, примерно $F_{40} = 102,334,155!$

Стъпки: $2^{0.694n} \approx (1.6)^n$

Рекурсивно дърво на Фибоначи

- **fib(n)** прави около **fib(n)** рекурсивни извиквания
- Една стойност се изчислява много много пъти!



Въведение в алгоритмите. Сложност на алгоритъм



Въпроси?



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni
Foundation

