

## Кратки записки на първите 6 упражнения по ДАА (гр. I)

Това са кратки записки на първите 6 упражнения по ДАА на гр. I. Въпреки, че пише „кратки“, някои задачи са решени подробно. Също така е възможно е някои от разгледаните тук примери да не са били разглеждани на упражнение.

### Упражнение I

*Def:* Ще казваме, че функцията  $f: \mathbb{R}^+ \rightarrow \mathbb{R}$  е *асимптотично неотрицателна*, ако:

$$\exists n_0 \in \mathbb{N} \forall n \geq n_0: f(n) \geq 0$$

Ако неравенството е строго, ще казваме, че  $f$  е *асимптотично положителна*.

*Def:* Нека  $g$  е асимптотично неотрицателна функция. Въвеждаме следните класове от функции (ще ги наричаме *асимптотични нотации*):

$$O(g) = \{f \mid \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq f(n) \leq c \cdot g(n)\}$$

$O(g)$  е множеството от всички функции, които растят *асимптотично не по-бързо* от  $g$ . Ще записваме това като  $f \in O(g)$ ,  $f = O(g)$  или  $f \preceq g$ .

$$\Omega(g) = \{f \mid \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq c \cdot g(n) \leq f(n)\}$$

$\Omega(g)$  е множеството от всички функции, които растят *асимптотично не по-бавно* от  $g$ . Ще записваме това като  $f \in \Omega(g)$ ,  $f = \Omega(g)$  или  $f \succeq g$ .

$$o(g) = \{f \mid \forall c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq f(n) < c \cdot g(n)\}$$

$o(g)$  е множеството от всички функции, които растят *асимптотично по-бавно* от  $g$ . Ще записваме това като  $f \in o(g)$ ,  $f = o(g)$  или  $f \prec g$ .

$$\omega(g) = \{f \mid \forall c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq c \cdot g(n) < f(n)\}$$

$\omega(g)$  е множеството от всички функции, които растят *асимптотично по-бързо* от  $g$ . Ще записваме това като  $f \in \omega(g)$ ,  $f = \omega(g)$  или  $f \succ g$ .

$$\Theta(g) = \{f \mid \exists c_1 > 0 \exists c_2 > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$$

$\Theta(g)$  е множеството от всички функции, които растят *асимптотично еднакво* с  $g$ . Ще записваме това като  $f \in \Theta(g)$ ,  $f = \Theta(g)$  или  $f \asymp g$ .

Някои основни свойства на асимптотичните нотации:

**Свойство 1:** Всички са транзитивни: Ако  $f \sigma g$  и  $g \sigma h$ , то  $f \sigma h$ , за  $\sigma \in \{<, >, \leq, \geq, =\}$

**Свойство 2:**  $\Theta, O$  и  $\Omega$  са рефлексивни:  $f \sigma f$  за  $\sigma \in \{\leq, \geq, =\}$

**Свойство 3:**  $f \geq g$  и  $g \geq f \Leftrightarrow f = g$

**Свойство 4:**  $\Theta$  е симетрична:  $f = g \Rightarrow g = f$

**Свойство 5:**  $f \leq g \Leftrightarrow g \geq f$  и  $f < g \Leftrightarrow g > f$

Доказателствата на горните свойства са директно от дефинициите.

**Свойство 6:**  $\max\{f, g\} = f + g$

*Доказателство:*

Ясно е, че за достатъчно големи  $n$ :

$$\frac{1}{2}f(n) + \frac{1}{2}g(n) \leq \max\{f(n), g(n)\} \leq f(n) + g(n) \Rightarrow \max\{f, g\} = f + g$$

**Свойство 7:** За асимптотично положителни  $f$  и  $g$  е вярно, че:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f = o(g)$$

*Доказателство:*

За произволно  $\varepsilon > 0$  и достатъчно големи  $n$  имаме:

$$-\varepsilon < \frac{f(n)}{g(n)} < \varepsilon \Leftrightarrow 0 \leq f(n) < \varepsilon \cdot g(n)$$

, което е достатъчно за еквивалентността.

**Свойство 8:** За асимптотично положителни  $f$  и  $g$  е вярно, че:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c > 0 \Rightarrow f = \Theta(g)$$

*Доказателство:*

За фиксирано  $c > \varepsilon > 0$ , и достатъчно големи  $n$  имаме:

$$c - \varepsilon < \frac{f(n)}{g(n)} < c + \varepsilon \Rightarrow 0 \leq (c - \varepsilon) \cdot g(n) \leq f(n) \leq (c + \varepsilon) \cdot g(n)$$

, което е достатъчно за свойството.

*Забележка:* Обратната посока на твърдението не е вярна.

Нека  $f(n) = (2 + \sin n) \cdot n$ , а  $g(n) = n$

Очевидно  $\forall n \geq 1: n \leq (2 + \sin n) \cdot n \leq 3n \Rightarrow f = \Theta(g)$ , но от друга страна границата:

$$\lim_{n \rightarrow \infty} \frac{(2 + \sin n) \cdot n}{n}$$

изобщо не съществува.

**Означение 1:** Със символът  $\lg n$  ще обозначаваме двоичен логаритъм, а не десетичен!

**Означение 2:** Тъй като  $\log_a n$  и  $\log_b n$  за  $a, b > 1$  се различават само с положителна константа:

$$\log_a n = \frac{\log_b n}{\log_b a}$$

, много често няма да обозначаваме основата. Например:  $\Theta(\lg n) = \Theta(\log n)$ .

**Свойство 9:** Нека  $f$  и  $g$  са асимптотично положителни и  $a > 1$ . (**Това се ползва много!**)

Ако  $f < g$  и  $g$  расте неограничено, то  $a^f < a^g$

Ако  $\log_a f < \log_a g$  и  $g$  расте неограничено, то  $f < g$

*Доказателство:*

Ключовото наблюдение за първото твърдение е, че  $g - f$  расте неограничено.

В противен случай:

$$1 = \lim_{n \rightarrow \infty} \frac{g(n) - f(n) + f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{g(n) - f(n)}{g(n)} + \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Сега:

$$\lim_{n \rightarrow \infty} \frac{a^{f(n)}}{a^{g(n)}} = \lim_{n \rightarrow \infty} \frac{1}{a^{g(n)-f(n)}} = 0$$

Второто свойство следва от първото.

*Забележка:* Тук е от особено значение, че  $g$  расте неограничено! Да разгледаме следния пример:

$$f(n) = \frac{1}{n}, \quad g(n) = 1 - \frac{1}{n}$$

Лесно се проверява, че:  $f < g$ , но  $2^f \approx 2^g$ .

**Свойство 10:**

$$\forall a > 1 \forall t > 0 \forall \varepsilon > 0: \log_a^t n < n^\varepsilon$$

*Доказателство:*

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{n^\varepsilon} = \lim_{x \rightarrow \infty} \frac{\log_a x}{x^\varepsilon}$$

От правилото на Лопитал имаме:

$$\lim_{x \rightarrow \infty} \frac{\log_a x}{x^\varepsilon} = \lim_{x \rightarrow \infty} \frac{1}{x \ln a \frac{\varepsilon}{x^\varepsilon} x^{\varepsilon-1}} = \lim_{x \rightarrow \infty} \frac{1}{\varepsilon \ln a x^\varepsilon} = 0$$

Сега:

$$\lim_{x \rightarrow \infty} \left( \frac{\log_a n}{n^\varepsilon} \right)^t = 0^t = 0 \Rightarrow \log_a^t n < n^\varepsilon$$

**Пример 1:**

Нека  $p(x) = \alpha_0 x^k + \alpha_1 x^{k-1} + \dots + \alpha_k$  е полином от степен  $k$  с  $\alpha_0 > 0$ . Тогава  $p(n) \asymp n^k$ .

*Доказателство:*

$$\lim_{n \rightarrow \infty} \frac{p(n)}{n^k} = \lim_{n \rightarrow \infty} \alpha_0 + \frac{\alpha_1}{n} + \dots + \frac{\alpha_k}{n^k} = \alpha_0 > 0 \Rightarrow p(n) \asymp n^k$$

**Пример 2:**

За фиксирано  $k \in \mathbb{N}$  е вярно, че:

$$\binom{n}{k} \asymp n^k$$

*Доказателство:*

$$\begin{aligned} \lim_{n \rightarrow \infty} \binom{n}{k} \cdot \frac{1}{n^k} &= \lim_{n \rightarrow \infty} \frac{n \cdot (n-1) \dots (n-k+1)}{k! \cdot n^k} = \lim_{n \rightarrow \infty} 1 \cdot \left(1 - \frac{1}{n}\right) \dots \left(1 - \frac{k-1}{n}\right) \cdot \frac{1}{k!} = \frac{1}{k!} \Rightarrow \\ &\Rightarrow \binom{n}{k} \asymp n^k \end{aligned}$$

**Пример 3:**

$$(n+1)^n \asymp n^n$$

*Доказателство:*

$$\lim_{n \rightarrow \infty} \frac{(n+1)^n}{n^n} = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e \Rightarrow (n+1)^n \asymp n^n$$

**Пример 4:**

Не всеки две асимптотично неотрицателни функции са асимптотично сравними.

*Доказателство:*

Нека:

$$g(n) = n, \quad f(n) = \begin{cases} 1 & , n \text{ е четно} \\ n^2 & , n \text{ е нечетно} \end{cases}$$

Ако допуснем, че  $f = O(g)$ , ще имаме:

$$\exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq f(n) \leq c \cdot g(n)$$

, което не е вярно за нечетни  $n > c$  ( $0 \leq n^2 \leq c \cdot n$ ).

Ако допуснем, че  $g = O(f)$ , ще имаме:

$$\exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq g(n) \leq c \cdot f(n)$$

, което не е вярно за четни  $n > c$  ( $0 \leq n \leq c$ ).

Щом  $f$  и  $g$  са несравними по  $O$ , те не са сравними изобщо.

### Пример 5:

Възможно е  $f = O(g)$ , без да е вярно нито едно от  $f = o(g)$  и  $f = \Theta(g)$ .

*Доказателство:*

Нека:

$$g(n) = n, \quad f(n) = \begin{cases} n & , n \text{ е четно} \\ \frac{1}{n} & , n \text{ е нечетно} \end{cases}$$

Вярно е, че  $f = O(g)$ , тъй като  $\forall n \geq 1: 0 \leq f(n) \leq g(n)$

Ако допуснем, че  $f = o(g)$ , ще имаме:

$$\forall c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq f(n) < c \cdot g(n)$$

, което не е вярно за четни  $n$  и  $c \leq 1$  ( $0 \leq n < c \cdot n$ ).

Ако допуснем, че  $f = \Theta(g)$ , ще имаме:

$$\exists c_1 > 0 \exists c_2 > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0: 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

, което не е вярно за нечетни  $n > \frac{1}{\sqrt{c_1}}$  ( $0 \leq c_1 \cdot n \leq \frac{1}{n}$ ).

## Упражнение II

Апроксимация на Стирлинг:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \theta\left(\frac{1}{n}\right)\right)$$

Пример 1:

$$\lg n! \approx n \lg n$$

Доказателство:

От апроксимацията на Стирлинг имаме:

$$\lg n! \approx \lg \left( \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \right) = \lg \sqrt{2\pi n} + n \lg n - n \lg e \approx n \lg n$$

Пример 2:

$$\binom{2n}{n} \approx \frac{4^n}{\sqrt{n}}$$

Доказателство:

От апроксимацията на Стирлинг имаме:

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \approx \frac{\sqrt{4\pi n} \left(\frac{2n}{e}\right)^{2n}}{2\pi n \left(\frac{n}{e}\right)^{2n}} = \sqrt{\frac{1}{\pi n}} 4^n \approx \frac{4^n}{\sqrt{n}}$$

Задача 1: Докажете, че  $1 < \lg \lg n < \lg n < n < n \lg n < n^2 < n^3 < 2^n < n! < n^n$

1.

$$\lim_{n \rightarrow \infty} \frac{1}{\lg \lg n} = 0 \Rightarrow 1 < \lg \lg n$$

2.

$$\lim_{n \rightarrow \infty} \frac{\lg \lg n}{\lg n} = \lim_{x \rightarrow \infty} \frac{\lg \lg x}{\lg x}$$

От правилото на Лопитал имаме:

$$\lim_{x \rightarrow \infty} \frac{\lg \lg x}{\lg x} = \lim_{x \rightarrow \infty} \frac{1}{\lg x} \cdot \frac{(\lg x)'}{(\lg x)'} = 0 \Rightarrow \lg \lg n < \lg n$$

3.

$$\lim_{n \rightarrow \infty} \frac{\lg n}{n} = \lim_{x \rightarrow \infty} \frac{\lg x}{x}$$

От правилото на Лопитал имаме:

$$\lim_{x \rightarrow \infty} \frac{\lg x}{x} = \lim_{x \rightarrow \infty} \frac{1}{x} = 0 \Rightarrow \lg n < n$$

4.

$$\lim_{n \rightarrow \infty} \frac{n}{n \lg n} = 0 \Rightarrow n < n \lg n$$

5.

$$\lim_{n \rightarrow \infty} \frac{n \lg n}{n^2} = 0 \Rightarrow n \lg n < n^2$$

6.

$$\lim_{n \rightarrow \infty} \frac{n^2}{n^3} = 0 \Rightarrow n^2 < n^3$$

7.

Тъй като  $\lg(n^3) = 3 \lg n < n \lg 2 = \lg(2^n)$ , то  $n^3 < 2^n$

8.

Тъй като  $\lg(2^n) = n \lg 2 < n \lg n \asymp \lg(n!)$ , то  $2^n < n!$

9.

Тъй като  $\forall n \in \mathbb{N}: 0 < n! \leq n^{n-1}$ , то:

$$\forall n \in \mathbb{N}: 0 < \frac{n!}{n^n} \leq \frac{1}{n}$$

От лемата за „двамата полицаи“ имаме:

$$\lim_{n \rightarrow \infty} \frac{n!}{n^n} = 0 \Rightarrow n! < n^n$$

**Задача 2:** Докажете, че  $\sqrt[n]{n} \asymp 1$

*Доказателство:*

Използваме следната лема:

$$\lim_{x \rightarrow \infty} f(x) = a > 0 \Leftrightarrow \lim_{x \rightarrow \infty} \ln f(x) = \ln a$$

Сега имаме:

$$\lim_{n \rightarrow \infty} \ln \sqrt[n]{n} = \lim_{n \rightarrow \infty} \frac{\ln n}{n} = 0 \Rightarrow \lim_{n \rightarrow \infty} \frac{\sqrt[n]{n}}{1} = e^0 = 1 \Rightarrow \sqrt[n]{n} \asymp 1$$

**Задача 3:** Подредете в асимптотично нарастващ ред функциите:

$$\sqrt{2}^{\lg n}, \quad n^3, \quad n!, \quad (\lg n)!, \quad \lg^2 n, \quad \lg n!, \quad 2^{2^n}, \quad n^{\frac{1}{\lg n}}, \quad \ln \ln n, \quad \left(\frac{3}{2}\right)^n$$

$$n \cdot 2^n, \quad 4^{\lg n}, \quad (n+1)!, \quad \sqrt{\lg n}, \quad 2^{\sqrt{2 \lg n}}, \quad n^{\lg \lg n}, \quad \ln n, \quad 2^{\lg n}, \quad (\lg n)^{\lg n}$$

Правилният ред е:

$$n^{\frac{1}{\lg n}} < \ln \ln n < \sqrt{\lg n} < \ln n < \lg^2 n < 2^{\sqrt{2 \lg n}} < \sqrt{2}^{\lg n} < 2^{\lg n} < \lg n! < 4^{\lg n} < n^3 < (\lg n)! <$$

$$< (\lg n)^{\lg n} \asymp n^{\lg \lg n} < \left(\frac{3}{2}\right)^n < n \cdot 2^n < n! < (n+1)! < 2^{2^n}$$

Решение:

1.

$$n^{\frac{1}{\lg n}} = n^{\log_n 2} = 2 < \ln \ln n$$

2.

$$\ln \ln n < \sqrt{\lg n}$$

, тъй като  $\sqrt{\lg n} \asymp \sqrt{\ln n}$  и  $\ln \ln n < \sqrt{\ln n}$ , подобно на  $\ln n < \sqrt{n}$

3.

$$\sqrt{\lg n} \asymp \sqrt{\ln n} < \ln n, \text{ тъй като } 1 < \sqrt{\ln n}$$

4.

$$\ln n < \lg^2 n, \text{ тъй като } \ln n < \ln^2 n \asymp \lg^2 n$$

5.

$$\lg^2 n < 2^{\sqrt{2 \lg n}}$$

Тъй като  $\lg(\lg^2 n) = 2 \lg \lg n < \sqrt{2 \lg n} \lg 2 = \lg(2^{\sqrt{2 \lg n}})$ , то  $\lg^2 n < 2^{\sqrt{2 \lg n}}$

6.

$$2^{\sqrt{2 \lg n}} < \sqrt{2}^{\lg n}$$

Тъй като  $\lg(2^{\sqrt{2 \lg n}}) = \sqrt{2 \lg n} \lg 2 < \frac{1}{2} \lg n = \lg(\sqrt{n}) = \lg \sqrt{2}^{\lg n}$ , то  $2^{\sqrt{2 \lg n}} < \sqrt{2}^{\lg n}$

7.

$$\sqrt{2}^{\lg n} = \sqrt{n} < n = 2^{\lg n}$$

8.

$$2^{\lg n} = n < n \lg n \asymp \lg n!$$

9.

$$\lg n! \asymp n \lg n < n^2 = 4^{\lg n}$$

10.

$$4^{\lg n} = n^2 < n^3$$

11.

$$n^3 < (\lg n)!$$

Тъй като  $\lg n^3 = 3 \lg n < \lg n \lg \lg n \asymp \lg(\lg n)!$ , то  $n^3 < (\lg n)!$

12.

$$(\lg n)! < (\lg n)^{\lg n}, \text{ подобно на } n! < n^n$$



13.

$$(\lg n)^{\lg n} = n^{\lg \lg n}$$

Това е заради свойството на логаритъма:  $a^{\log_b c} = c^{\log_b a}$

14.

$$n^{\lg \lg n} < \left(\frac{3}{2}\right)^n$$

Тъй като  $\lg(n^{\lg \lg n}) = \lg \lg n \cdot \lg n < \lg^2 n < n \lg \frac{3}{2} = \lg \left(\frac{3}{2}\right)^n$ , то  $n^{\lg \lg n} < \left(\frac{3}{2}\right)^n$

15.

$$\left(\frac{3}{2}\right)^n < 2^n < n \cdot 2^n$$

16.

$$n \cdot 2^n < n!$$

Тъй като  $\lg n \cdot 2^n = \lg n + n \lg 2 < n \lg n = \lg(n!)$ , то  $n \cdot 2^n < n!$

17.

$$n! < (n+1)!$$

$$\lim_{n \rightarrow \infty} \frac{n!}{(n+1)!} = \lim_{n \rightarrow \infty} \frac{1}{n+1} = 0 \Rightarrow n! < (n+1)!$$

18.

$$(n+1)! < 2^{2^n}$$

Тъй като  $\lg(n+1)! = (n+1) \cdot \lg(n+1) < 2^n \lg 2 = \lg(2^{2^n})$ , то  $(n+1)! < 2^{2^n}$

## Упражнение III

**Пример 1:** Даденият алгоритъм връща  $2^n$ :

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

*Инварианта на цикъла:* При всяко  $k$ -то достигане на ред l:  $s = 2^{k-1}$ .

*Доказателство:*

База: При първото достигане:  $i = 0, s = 2^0 = 1$  - вярно.

Поддръжка: Нека е вярно за някое  $k$ -то достигане, което не е последното:

$$i = k - 1, s = 2^{k-1}$$

В тялото на цикъла имаме  $s = s * 2$ , така че на  $k + 1$ -вото достигане ще имаме:  
 $i = k, s = 2^{k-1} \cdot 2 = 2^k$  - твърдението отново е вярно.

Терминация: При  $n + 1$ -вото достигане имаме:  $i = n, s = 2^n$ .

В този момент цикълът приключва.

След цикъла имаме return s, което връща стойността на s или  $2^n$ .

**Пример 2:** Даденият алгоритъм връща сумата на елементите на масива  $a[n]$ :

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

*Инварианта на цикъла:*

В началото на всяка итерация на цикъла s съдържа  $a[0] + \dots + a[i - 1]$ .

**Пример 3:** Коректност на Selection Sort

```
Selection_sort(int a[n])
{
    for (int i = 0; i < n - 1; i++)
    {
        int m = i;
        for (int j = i + 1; j < n; j++) if (a[j] < a[m]) m = j;
        swap(a[i], a[m]);
    }
}
```

*Инварианта на вътрешния цикъл:*

В началото на всяка итерация на вътрешния цикъл  $a[m] = \min\{a[i], \dots, a[j - 1]\}$ .

*Инварианта на външния цикъл:*

В началото на всяка итерация на външния цикъл елементите  $a[0], \dots, a[i - 1]$  са сортирани и в тях се съдържат  $i$ -тите най-малки елементи на масива.

**Пример 4:** Бързо вдигане на степен

```
int exp_by_sqr(int x, int n)
{
    if (n == 0) return 1;
    if (n is even) return exp_by_sqr(x * x, n / 2);
    return x * exp_by_sqr(x * x, n / 2);
}
```

*Твърдение:* Алгоритъмът връща  $x^n$ .

Проверява се непосредствено като се разгледат всички случаи.

**Пример 5:** Една задача на Кралчев.

Дадена е кутия с 53 сини и 42 жълти пьонки. Извън кутията има неограничен запас от жълти пьонки. Разглеждаме следния алгоритъм:

```
Alg()
{
    Докато (не остане само 1 пьонка в кутията)
    {
        Извади 2 случайни пьонки.
        Ако са с еднакъв цвят - добави 1 жълта, иначе върни синята.
    }
}
```

Какъв ще е цветът на последната пьонка в кутията?

*Инварианта на цикъла:*

В началото на всяка  $k$ -та итерация на цикъла броят пьонки в кутията е  $95 - k + 1$ , като броят на сините пьонки е нечетен.

**Пример 6:** Алгоритъм на Кадан за подмасив с максимална сума.

```
Kadane(int a[n])
{
    int c = 0, m = 0;
    for (int i = 0; i < n; i++)
    {
        if (a[i] + c > 0) c = a[i] + c; else c = 0;
        if (m > c) m = c;
    }
    return m;
}
```

*Инварианта на цикъла:*

В началото на всяка итерация на цикъла  $c$  съдържа най-голямата сума, на подмасив завършващ в  $a[i - 1]$ , а  $m$  съдържа най-голямата сума сред тези, завършващи в  $a[0], \dots, a[i - 1]$ .

*Забележка:* Този алгоритъм намира и празни подмасиви (ако всички елементи са отрицателни).

Може лесно да се модофицира да работи за непразни:

```
Kadane2(int a[n])
{
    int c = a[0], m = a[0];
    for (int i = 1; i < n; i++)
    {
        if (a[i] + c > a[i]) c = a[i] + c; else c = a[i];
        if (m > c) m = c;
    }
    return m;
}
```

### Упражнение IV

Някои важни суми:

$$\sum_{i=1}^n 1 = n, \quad \sum_{i=a}^b 1 = b - a + 1, \quad \sum_{i=1, i=i+k}^n 1 = \left\lfloor \frac{n}{k} \right\rfloor \approx \frac{n}{k}$$
$$\sum_{i=1}^n i = \frac{n(n+1)}{2} = \Theta(n^2), \quad \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$$
$$\sum_{i=1}^n i^\alpha = \begin{cases} \Theta(n^{\alpha+1}) & , \alpha > -1 \\ \Theta(\log n) & , \alpha = -1 \\ \Theta(1) & , \alpha < -1 \end{cases}$$

Тези отношения могат да бъдат изведени по различни начини. Първите 5 могат да се докажат лесно по индукция или да се проверят непосредствено. Шестото е следствие от интегралния критерий за суми.

В следващите примери условието е да се намери сложността по време на дадения фрагмент.

**Пример 1:**

```
for (int i = 1; i <= n; i++) print("a");
```

Времето, за което се изпълнява цикълът може да се изрази със сумата:

$$\sum_{i=1}^n c = c \sum_{i=1}^n 1 = \Theta(n)$$

Идеята е, че той има  $n$  итерация като на всяка от тях извършва една и съща константна работа със сложност  $c > 0$ . В следващите примери ще отбелязваме това с 1, вместо с  $c$ , тъй като в този случай константата не оказва влияние на асимптотичното поведение на сумите.

### Пример 2:

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++) print("a");
```

Времето, за което се изпълняват циклите може да се изрази със сумата:

$$\sum_{i=1}^n \sum_{j=1}^n 1 = \sum_{i=1}^n n = n^2 = \Theta(n^2)$$

### Пример 3:

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= i; j++) print("a");
```

Времето, за което се изпълняват циклите може да се изрази със сумата:

$$\sum_{i=1}^n \sum_{j=1}^i 1 = \sum_{i=1}^n i = \Theta(n^2)$$

### Пример 4:

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j += i) print("a");
```

Времето, за което се изпълняват циклите може да се изрази със сумата:

$$\sum_{i=1}^n \sum_{j=1, j=j+i}^n 1 \approx \sum_{i=1}^n \frac{n}{i} = n \sum_{i=1}^n \frac{1}{i} = \Theta(n \log n)$$

### Пример 5:

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
        if (i == j) for (int k = 1; k <= n; k++) print("a");
```

Първите два вложени цикъла определят време за изпълнение  $n^2$ . Третият цикъл се изпълнява само в случаите, когато  $i = j$ , а те са точно  $n$  на брой ( $i = j = 1, i = j = 2$  и така нататък). Той от своя страна е с време  $n$ , така че за него имаме общо време  $n^2$ .

Цялото време на изпълнение се определя от времето на двата вложени цикъла и случаите, когато се изпълнява третият цикъл, т.е.  $n^2 + n^2 = 2n^2 = \Theta(n^2)$ .

### Пример 6:

```

for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j+=i)
        for (int k = 1; k <= n; k += i) print("a");

```

Времето, за което се изпълняват циклите може да се изрази със сумата:

$$\sum_{i=1}^n \sum_{j=1, j=j+i}^n \sum_{k=1, k=k+i}^n 1 \approx \sum_{i=1}^n \sum_{j=1, j=j+i}^n \frac{n}{i} = \sum_{i=1}^n \left( \frac{n}{i} \cdot \sum_{j=1, j=j+i}^n 1 \right) \approx \sum_{i=1}^n \frac{n^2}{i^2} = n^2 \cdot \sum_{i=1}^n \frac{1}{i^2} = \Theta(n^2)$$

### Пример 7: Сложност на *BuildHeap*

Наблюдение: За  $|x| < 1$  имаме:

$$\frac{x}{(1-x)^2} = x \left( \frac{1}{1-x} \right)' = x \cdot \left( \sum_{n=0}^{\infty} x^n \right)' = \sum_{n=1}^{\infty} nx^n$$

Следният израз описва най-лошия случай за *BuildHeap* на пълна двойчна пирамида с височина  $h$  и  $n = 2^{h+1} - 1$  елемента:

$$S(n) = \sum_{i=1}^h 2^{h-i} \cdot i = 2^h \sum_{i=1}^h \frac{i}{2^i} \leq 2^h \sum_{i=1}^{\infty} \frac{i}{2^i} = 2^h \cdot 2 = O(2^h) = O(n)$$

Сложността  $O(n)$  в общия случай се съобразява от факта, че работата на алгоритъма в този случай е по-малка от работата за най-малката пълна двойчна пирамида с размер  $\geq n$ . Този размер е  $s \leq 2n$ , а търсената работа е  $O(s)$ , която ненадвишава  $O(2n) = O(n)$ .

Долната граница  $\Omega(n)$  идва от факта, че трябва да разгледаме всички елементи на пирамидата.

## Упражнение V

Условието на всички примери е да се намери сложността на рекурентното уравнение.

Ще считаме, че за достатъчно малки  $n$ ,  $T(n)$  се изпълнява за константно време.

### Пример 1:

$$T(n) = 4T(n-2) + n \cdot 2^n + 4 \cdot 3^n$$

От хомогенната част на това уравнение имаме:

$$x^2 - 4 = 0, x = \pm 2 \Rightarrow \{2, -2\}_m$$

От нехомогенната част имаме  $\{2, 2\}_m$  от  $n \cdot 2^n$  и  $\{3\}_m$  от  $4 \cdot 3^n$

Общото решение има вида:

$$T(n) = c_1 3^n + c_2 2^n + c_3 n 2^n + c_4 n^2 2^n + c_5 (-2^n) = \Theta(3^n), \text{ където } c_1 > 0$$

**Пример 2:**

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

От хомогенната част на това уравнение имаме:

$$x^2 - 2x + 1 = 0, x_{1,2} = 1 \Rightarrow \{1, 1\}_m$$

Общото решение има вида:

$$T(n) = c_1 1^n + c_2 n \cdot 1^n = \Theta(n), \text{ където } c_2 > 0$$

**Пример 3:**

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

С разписване изразите за  $T(n)$  получаваме:  $T(n) = T(0) + n = \Theta(n)$

**Пример 4:**

$$T(n) = T(n-1) + \frac{1}{n}$$

С разписване на изразите за  $T(n)$  получаваме:

$$T(n) = T(0) + \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \dots + \frac{1}{1} = T(0) + \sum_{i=1}^n \frac{1}{i} = \Theta(\ln n)$$

**Пример 5:**

$$T(n) = 2T(n-1) + \frac{1}{n}$$

С разписване на изразите за  $T(n)$  получаваме:

$$T(n) = 2^n T(0) + \frac{1}{n} + \frac{2}{n-1} + \frac{4}{n-2} + \dots + \frac{2^{n-1}}{1} = 2^n \cdot T(0) + 2^n \sum_{i=1}^n \frac{1}{i \cdot 2^i} = \Theta(2^n)$$

, тъй като:

$$\sum_{i=1}^n \frac{1}{i \cdot 2^i} \leq \sum_{i=1}^n \frac{1}{2^i} \leq \sum_{i=1}^{\infty} \frac{1}{2^i} = 1 \Rightarrow 2^n \sum_{i=1}^n \frac{1}{i \cdot 2^i} \leq 2^n$$

**Пример 6:**

$$T(n) = \frac{n}{n+1} T(n-1) + 1$$

С разписване на изразите за  $T(n)$  получаваме:

$$\begin{aligned} T(n) &= \frac{1}{n+1} T(0) + 1 + \frac{n}{n+1} + \frac{n-1}{n+1} + \dots + \frac{2}{n+1} = \frac{1}{n+1} \cdot T(0) + \frac{1}{n+1} \sum_{i=2}^{n+1} i = \\ &= \frac{1}{n+1} \cdot T(0) + \frac{1}{n+1} \left( \frac{(n+1)(n+2)}{2} - 1 \right) = \Theta(n) \end{aligned}$$

**Пример 7:** Анализ на средния случай на *QuickSort*

$$Q(n) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} [Q(i) + Q(n-1-i)] + c \cdot n = \frac{2}{n} \cdot \sum_{i=0}^{n-1} Q(i) + c \cdot n$$

Оттук получаваме:

$$n \cdot Q(n) = 2 \cdot \sum_{i=0}^{n-1} Q(i) + c \cdot n^2$$

, откъдето

$$(n-1) \cdot Q(n-1) = 2 \cdot \sum_{i=0}^{n-2} Q(i) + c \cdot (n-1)^2$$

Изваждаме двата израза и получаваме:

$$Q(n) = \frac{n+1}{n} Q(n-1) + 2c - \frac{c}{n}$$

С разписване на изразите за  $Q(n)$  получаваме:

$$Q(n) = (n+1)Q(0) + 2c(n+1) \cdot \sum_{i=2}^{n+1} \frac{1}{i} - c(n+1) \cdot \sum_{i=2}^{n+1} \frac{1}{i \cdot (i-1)}$$

, откъдето

$$Q(n) = (n+1)Q(0) + 2c(n+1) \sum_{i=2}^{n+1} \frac{1}{i} - cn = \Theta(n \log n)$$

**Пример 8:**

$$T(n) = 2T(\sqrt{n}) + 1$$

Представяме  $n$  във вида  $2^{2^m}$  ( $m = \lg \lg n$ )

Сега имаме:

$$S(m) = T(2^{2^m}) = 2T(2^{2^{m-1}}) + 1 \text{ или } S(m) = 2S(m-1) + 1$$

Сложността на това уравнение е  $\Theta(2^m)$ , което спрямо  $n$  е  $\Theta(2^{\lg \lg n}) = \Theta(\lg n)$

Какво става, ако  $n$  няма вида  $2^{2^m}$  за цяло число  $m$ ?

Наблюдението е, че рекурентното уравнение за  $T$  е растящо.

За достатъчно големи  $m$ , нека  $x \in [m, m+1]$ .

Сега:

$$\frac{1}{2} \cdot c_1 \cdot 2^x \leq \frac{1}{2} c_1 \cdot 2^{m+1} = c_1 \cdot 2^m \leq T(2^{2^m}) \leq T(2^{2^x}) \leq T(2^{2^{m+1}}) \leq c_2 \cdot 2^{m+1} = 2 \cdot c_2 \cdot 2^m \leq 2 \cdot c_2 \cdot 2^x$$

, откъдето за достатъчно големи  $n$ :

$$\frac{1}{2} \cdot c_1 \cdot \lg n \leq T(n) \leq 2 \cdot c_2 \lg n$$



**Пример 9:**

$$T(n) = T(\sqrt{n}) + 1$$

Представяме  $n$  във вида  $2^{2^m}$  ( $m = \lg \lg n$ )

Сега имаме:

$$S(m) = T(2^{2^m}) = T(2^{2^{m-1}}) + 1 \text{ или } S(m) = S(m-1) + 1$$

Сложността на това уравнение е  $\Theta(m)$ , което спрямо  $n$  е  $\Theta(\lg \lg n)$

**Пример 10:** Да се намери сложността на следната рекурсивна програма:

```
A(int n)
{
    int s = 0;
    for (int i = 1; i < n; i++) s = s + 2 * A(i) + 1;
    return s;
}
```

*Решение:*

На всяка итерация на цикъла, програмата се обръща към себе си с по-малък аргумент. Умножението с 2 приемаме за константна операция, така че реално програмата върши само веднъж работата за този по-малък аргумент, а не два пъти!

Ако изразът беше  $s = s + a(i) + a(i) + 1$ , то резултатът щеше да е същият, но сложността щеше да е друга, тъй като в този случай програмата реално щеше да върши два пъти работата за по-малкия аргумент!

Сложността може да се опише със следното рекурентно уравнение:

$$T(n) = T(1) + c + T(2) + c + \dots + T(n-1) + c = \sum_{i=1}^{n-1} T(i) + (n-1)c$$

$c > 0$  показва допълнителната константна работа, която върши алгоритъмът.

Аналогично за  $T(n-1)$  имаме:

$$T(n-1) = \sum_{i=1}^{n-2} T(i) + (n-2)c$$

, откъдето:

$$T(n) - T(n-1) = \sum_{i=1}^{n-1} T(i) + (n-1)c - \sum_{i=1}^{n-2} T(i) - (n-2)c = T(n-1) + c \Rightarrow$$

$$\Rightarrow T(n) = 2 \cdot T(n-1) + c$$

Това рекурентно уравнение има общо решение:  $T(n) = c_1 \cdot 2^n + c_2 = \Theta(2^n)$ , където  $c_1 > 0$

## Упражнение VI

**Теорема:** (Мастър теорема)

За  $a \geq 1$ ,  $b > 1$  и  $f(n)$  - положителна е дадено рекурентното уравнение:

$$T(n) = a.T\left(\frac{n}{b}\right) + f(n)$$

*Случай 1:* Ако  $f(n) = O(n^{\log_b a - \varepsilon})$  за някое  $\varepsilon > 0$ , то  $T(n) = \Theta(n^{\log_b a})$

*Случай 2:* Ако  $f(n) = \Theta(n^{\log_b a})$ , то  $T(n) = \Theta(n^{\log_b a} \lg n)$

*Случай 3:* Ако  $f(n) = \Omega(n^{\log_b a + \varepsilon})$  за някое  $\varepsilon > 0$  и

$$\exists c \in (0, 1) \exists n_0 \in \mathbb{N} \forall n \geq n_0: a.f\left(\frac{n}{b}\right) \leq c.f(n)$$

, то  $T(n) = \Theta(f(n))$

Втората част от *Случай 3* се нарича *условие за регулярност*.

Доказателство можете да намерите на страници 97 - 106 от Т. Cormen, С. Leiserson - Introduction to Algorithms, Third Edition.

Условието на следващите примери е да се намери сложността на рекурентното уравнение:

**Пример 1:**

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

$$\text{Имаме: } n = O(n^{2-0.1}) = O(n^{\log_2 4-0.1}) \xrightarrow{\text{MTh 1}} T(n) = \Theta(n^2)$$

**Пример 2:**

$$T(n) = 4T\left(\frac{n}{\sqrt{2}}\right) + n^3$$

$$\text{Имаме: } n^3 = O(n^{4-0.1}) = O(n^{\log_{\sqrt{2}} 4-0.1}) \xrightarrow{\text{MTh 1}} T(n) = \Theta(n^4)$$

**Пример 3:**

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$\text{Имаме: } n = \Theta(n) = \Theta(n^{\log_2 2}) \xrightarrow{\text{MTh 2}} T(n) = \Theta(n \lg n)$$

**Пример 4:**

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

$$\text{Имаме: } 1 = \Theta(1) = \Theta(n^{\log_2 1}) \xrightarrow{\text{MTh 2}} T(n) = \Theta(\lg n)$$

**Пример 5:**

$$T(n) = 2T\left(\frac{n}{8}\right) + n$$

Имаме:  $n = \Omega(\sqrt{n}) = \Omega\left(n^{\log_8 2 + \frac{1}{6}}\right)$

Освен това:

$$\forall n \geq 1: 2 \cdot \frac{n}{8} \leq \frac{1}{4} \cdot n$$

От *MTh 3*  $\Rightarrow T(n) = \Theta(n)$

**Пример 6:**

$$T(n) = 3T\left(\frac{n}{9}\right) + n \lg n$$

Имаме:  $n \lg n = \Omega\left(n^{\frac{5}{6}}\right) = \Omega\left(n^{\log_9 3 + \frac{1}{3}}\right)$

Освен това:

$$\forall n \geq 1: 3 \cdot \frac{n}{9} \cdot \lg \frac{n}{9} \leq \frac{1}{3} n \lg n$$

От *MTh 3*  $\Rightarrow T(n) = \Theta(n \lg n)$

**Пример 7:**

$$T(n) = 8T\left(\frac{n}{4}\right) + n \lg n$$

Имаме:  $n \lg n = O\left(n^{\frac{3}{2}-0.1}\right) = O\left(n^{\log_4 8-0.1}\right) \xrightarrow{MTh 1} T(n) = \Theta\left(n^{\frac{3}{2}}\right)$

**Пример 8:**

$$T(n) = 2\sqrt{2}T\left(\frac{n}{\sqrt{2}}\right) + n^3$$

Имаме:  $n^3 = \Theta(n^3) = \Theta\left(n^{\log_{\sqrt{2}} 2\sqrt{2}}\right) \xrightarrow{MTh 2} T(n) = \Theta(n^3 \lg n)$

**Пример 9:**

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2\sqrt{n}$$

Имаме:  $n^2\sqrt{n} = \Omega\left(n^{2+0.1}\right) = \Omega\left(n^{\log_2 4+0.1}\right)$

Освен това:

$$\forall n \geq 1: 4 \cdot \frac{n^2\sqrt{n}}{4\sqrt{2}} \leq \frac{1}{\sqrt{2}} n^2\sqrt{n}$$

От *MTh 3*  $\Rightarrow T(n) = \Theta(n^2\sqrt{n})$

**Частен случай на Мастър теоремата:**

За  $a \geq 1$ ,  $b > 1$  и  $f(n)$  - положителна е дадено рекурентното уравнение:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Ако  $f(n) = \Theta(n^{\log_b a} \lg^t n)$  за някое  $t \geq 0$ , то  $T(n) = \Theta(n^{\log_b a} \lg^{t+1} n)$

**Пример 11:**

$$T(n) = 2T\left(\frac{n}{4}\right) + 2\sqrt{n} \lg^3 n$$

Имаме:

$$2\sqrt{n} \lg^3 n = \Theta(\sqrt{n} \lg^3 n) = \Theta(n^{\log_4 2} \lg^3 n)$$

От частния случай  $\Rightarrow T(n) = \Theta(\sqrt{n} \lg^4 n)$

**Пример 12:**

$$T(n) = T\left(\frac{n}{2}\right) + \lg n$$

Имаме:

$$\lg n = \Theta(\lg n) = \Theta(n^{\log_2 1} \lg n)$$

От частния случай  $\Rightarrow T(n) = \Theta(\lg^2 n)$

**Пример 13:**

$$T(n) = T\left(\frac{n}{2}\right) + \frac{1}{\lg n}$$

Тук не е приложим дори частният случай!

1.  $\frac{1}{\lg n} \neq O(n^{-\varepsilon})$ , тъй като  $\frac{1}{\lg n} > n^{-\varepsilon}$  за  $\varepsilon > 0$
2.  $\frac{1}{\lg n} \neq \Theta(1)$ , тъй като  $\frac{1}{\lg n} < 1$
3.  $\frac{1}{\lg n} \neq \Omega(n^\varepsilon)$ , тъй като  $\frac{1}{\lg n} < n^\varepsilon$  за  $\varepsilon > 0$
4.  $\frac{1}{\lg n} = (\lg n)^{-1}$

Нека  $n = 2^m$ .

Сега имаме:

$$S(m) = T(2^m) = T(2^{m-1}) + \frac{1}{m} = S(m-1) + \frac{1}{m}$$

Сложността на  $S(m)$  се изразява чрез сумата:

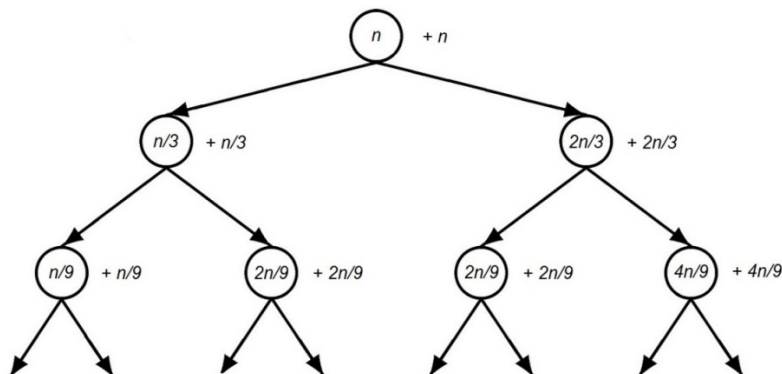
$$S(m) = S(0) + 1 + \frac{1}{2} + \dots + \frac{1}{m} = S(0) + \sum_{i=1}^m \frac{1}{i} \asymp \ln m \asymp \lg m = \lg \lg n$$

**Пример 14:** Да се реши следното рекурентно уравнение:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$

*Решение:*

Да разгледаме фрагмент от дървото на извод за  $T(n)$ :



Върховете на това дърво представляват стойностите на  $n$  за съответното рекурсивно извикване. Отстрани е написана допълнителната работа към всеки връх.

Височината на най-лявото листо е  $\log_3 n$ . До тази височина дървото е пълно. На всяко ниво до тази височина включително имаме сумарна допълнителна работа  $n$ , откъдето получаваме долна граница за сложността на уравнението -  $\Omega(n \log n)$ .

Можем да докажем, че  $\exists c \geq 10$ , такава че за достатъчно големи  $n$ :

$$T(n) \leq c \cdot n \cdot \lg n$$

Това се проверява непосредствено прилагайки индукционната хипотеза:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n \leq c \cdot \frac{n}{3} \cdot \lg \frac{n}{3} + c \cdot \frac{2n}{3} \cdot \lg \frac{2n}{3} + n = c \cdot n \cdot \lg n - cn \left( \lg 3 - \frac{1}{c} - \frac{2}{3} \right) \leq c \cdot n \cdot \lg n$$

В такъв случай  $T(n) = \Omega(n \log n)$  и  $T(n) = O(n \log n)$ , откъдето  $T(n) = \Theta(n \log n)$ .