Задача 1: Докажете по индукция, че рекурентното уравнение

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

има решение $T(n) \asymp n^{1+\lg n}$.

Решение: Първо ще докажем, че $T(n) \preceq n^{1+\lg n}$. Ще използваме засилване на твърдението: ще докажем, че съществуват положителни константи b, c, такива че

$$T(n) \leq c \cdot n \cdot n^{\lg n} - b \tag{1}$$

за всички достатъчно големи n. Индуктивното предположение е, че

$$T\left(\frac{n}{2}\right) \leq c \cdot \frac{n}{2} \cdot \left(\frac{n}{2}\right)^{\lg \frac{n}{2}} - b$$

Тогава

$$\begin{aligned}
T(n) &\leq n^2 \left(c \cdot \frac{n}{2} \cdot \left(\frac{n}{2}\right)^{\lg \frac{n}{2}} - b\right) + 1 \\
&= c \cdot \frac{n}{2} \cdot n^2 \cdot \frac{n^{\lg \frac{n}{2}}}{2^{\lg \frac{n}{2}}} - bn^2 + 1 \\
&= c \cdot \frac{n}{2} \cdot n^2 \cdot \frac{\frac{n^{\lg n}}{n}}{\frac{2^{\lg n}}{2}} - bn^2 + 1 \\
&= c \cdot \frac{n}{2} \cdot n^2 \cdot \frac{n^{\lg n}}{n} \cdot \frac{2}{2^{\lg n}} - bn^2 + 1 \\
&= c \cdot \frac{n}{2} \cdot n^2 \cdot \frac{n^{\lg n}}{n} \cdot \frac{2}{n} - bn^2 + 1 \\
&= c \cdot n \cdot n^{\lg n} - bn^2 + 1 \\
&= c \cdot n \cdot n^{\lg n} - b + (-bn^2 + 1 + b) \\
&\leq c \cdot n \cdot n^{\lg n} - b
\end{aligned}$$

ако $(-bn^2 + 1 + b) < 0$ за всички достатъчно големи n. Но $-bn^2+1+b$ е неограничено намаляваща функция за всяко положително b, когато n клони към безкрайност. Ясно е, че за $c = 1$ и $b = 1$ неравенстно (1) е изпълнено.

Сега ще докажем, че $T(n) \succeq n^{1+\lg n}$. Ще докажем, че съществува положителна константа d, такава че

$$T(n) \geq d \cdot n \cdot n^{\lg n} \tag{2}$$

за всички достатъчно големи n. Индуктивното предположение е, че

$$T\left(\frac{n}{2}\right) \geq d \cdot \frac{n}{2} \cdot \left(\frac{n}{2}\right)^{\lg \frac{n}{2}}$$

Тогава

$$
\begin{aligned}
T(n) &\geq n^2 \left(d \cdot \frac{n}{2} \cdot \left(\frac{n}{2}\right)^{\lg \frac{n}{2}} \right) + 1 \\
&= d \cdot \frac{n}{2} \cdot n^2 \cdot \frac{n^{\lg \frac{n}{2}}}{2^{\lg \frac{n}{2}}} + 1 \\
&= d \cdot \frac{n}{2} \cdot n^2 \cdot \frac{\frac{n^{\lg n}}{n}}{\frac{2^{\lg n}}{2}} + 1 \\
&= d \cdot \frac{n}{2} \cdot n^2 \cdot \frac{n^{\lg n}}{n} \cdot \frac{2}{2^{\lg n}} + 1 \\
&= d \cdot \frac{n}{2} \cdot n^2 \cdot \frac{n^{\lg n}}{n} \cdot \frac{2}{n} + 1 \\
&= d \cdot n \cdot n^{\lg n} + 1 \\
&\geq d \cdot n \cdot n^{\lg n}
\end{aligned}
$$

Тогава (2) е вярно за, да кажем, $d = 1$, за всички достатъчно големи n.

Задача 2: Разгледайте функцията `foo`, написана на `C`. Нека а е положително.

```c
1  int foo(int a) {
2    int i, x = 6, y = 1, z = 0;
3
4    for (i = 0; i < a; i++) {
5      z += y;
6      y += x;
7      x += 6;
8    }
9
10   return z;
11 }
```

foo.c

- Какво връща тя?
- Докажете това колкото можете по-формално и прецизно.

Решение: Пускаме програмата върху $a = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ и виждаме, че тя връща съответно $1, 8, 27, 64, 125, 216, 343, 512, 729, 1\,000$. Хипотезата е, че програмата връща a^3. Преди формалното доказателство да видим малко обяснения.

Променливата i става a при последното достигане на ред 4. Ако върнатото z е a^3, би трябвало при всяко достигане на ред 4, z да е i^3. Тогава z взема последователно стойностите 0, 1, 8, 27 и така нататък. Ако това е така, то y има смисъл на нарастването от даден точен куб към следващия. Тогава x би трябвало да има смисъл на нарастване на нарастването. Защо това става с добавяне на 6?

Да разгледаме редицата от точните кубове

0	1	8	27	64	125

Да си представим разликите (делтите) между съседните елементи, написани отдолу в червено.

0		1		8		27		64		125
	1		7		19		37		61	

Сега да си представим разликите между разликите (делтите на делтите), написани със синьо:

0		1		8		27		64		125
	1		7		19		37		61	
		6		12		18		24		

Ако продължим в същия дух с делтите на делтите на делтите в зелено, ще получим

0		1		8		27		64		125
	1		7		19		37		61	
		6		12		18		24		30
			6		6		6		6	

Ако продължим още надолу, следващият ред ще е само от нули и следващите редове ще са само от нули.

Същото нещо може да бъде изразено прецизно с оператора-разлика $\mathbf{\Delta}$. Ако $f(n)$ е функция, то $\mathbf{\Delta} f(n)$ е $f(n+1) - f(n)$. В случая $f(n) = n^3$, така че

$$\mathbf{\Delta} n^3 = (n+1)^3 - n^3 = 3n^2 + 3n + 1$$

И наистина, y взема стойностите 1, 7, 19, 37 и така нататък, като при всяко достигане на ред 4, y е точно $3i^2 + 3i + 1$.

Но $\mathbf{\Delta} f(n)$ е функция и можем да вземем нейното $\mathbf{\Delta}$. Това е $\mathbf{\Delta}^2 f(n)$: двукатното действие на оператора $\mathbf{\Delta}$ върху $f(n)$. В нашия случай,

$$\mathbf{\Delta}^2 n^3 = \mathbf{\Delta}(\mathbf{\Delta} n^3) = \mathbf{\Delta}(3n^2+3n+1) = 3(n+1)^2+3(n+1)+1-(3n^2+3n+1) = 6n+6$$

И наистина, x взема стойностите 6, 12, 18, 24 и така нататък, като при всяко достигане на ред 4, x е точно $6i + 6$.

Ако продължим в същия дух, получаваме $\mathbf{\Delta}^3 n^3 = 6$, $\mathbf{\Delta}^4 n^3 = 0$, $\mathbf{\Delta}^5 n^3 = 0$ и така нататък, но на тези редици няма съответни променливи в програмата. Забележете, че $\mathbf{\Delta}$ в някакъв смисъл съответства на прозводната в континуалния анализ, $\mathbf{\Delta}^2$ съответства на втората производна и така нататък. Примерно, производната на p^3 е $3p^2$, втората ѝ производна е $6p$, третата ѝ производна е 6 и всички производни нататък са нули.

Сега би трябвало интуитивно да е ясно защо програмата връща куба на числото-вход. Следва формалното доказателство.

Теорема 1 *За всяко положително* a*,* foo(a) *връща* a^3*.*

Доказателство: Инвариант на цикъла на редове 4–8 е следното твърдение:

> При всяко достигане на ред 4, променливата z съдържа i^3, променливата y съдържа 3i^2 + 3i + 1 и променливата x съдържа 6i + 6.

В базовия случай i е 0. Инвариантът става "z съдържа 0, променливата y съдържа 1 и променливата x съдържа 6", което е очевидно вярно предвид присвояванията на ред 2.

Да видим поддръжката. Да допуснем, че инвариантът е изпълнен при някое достигане на ред 4, което не е последното. В този момент имаме

$$z = i^3$$
$$y = 3i^2 + 3i + 1$$
$$x = 6i + 6$$

На ред 5 към z се добавя y. Съгласно предположението, новото z е $i^3 + 3i^2 + 3i + 1$. Но тогава новото z е $(i+1)^3$.

На ред 6 към y се добавя x. Съгласно предлоположението, новото y е $3i^2 + 3i + 1 + 6i + 6$. Но това е $3i^2 + 6i + 3 + 3i + 3 + 1 = 3(i^2 + 2i + 1) + 3(i+1) + 1$, което е $3(i+1)^2 + 3(i+1) + 1$.

На ред 7 към x се добавя 6. Съгласно предположението, новото x е $6i + 6 + 6 = 6(i + 1) + 6$.

Изпълнението отива на ред 4, където i бива инкрементирана. Изразено чрез новото i, в сила е

$$\begin{aligned} &z = i^3 \\ &y = 3i^2 + 3i + 1 \\ &x = 6i + 6 \end{aligned}$$

Инвариантът е доказан. При последното достигане на ред 4, очевидно i съдържа a. Тогава, съгласно инварианта, z съдържа a^3, което алгоритъмът връща. □

Задача 3: Разгледайте алгоритъма SomeAlg:

SomeAlg(A: масив от цели числа, ℓ, h: индекси в A)

1 **if** $\ell < h$
2   **if** $A[\ell] > A[h]$
3   swap($A[\ell], A[h]$)
4     $t \leftarrow \left\lfloor \frac{h-\ell+1}{3} \right\rfloor$
5   **if** $t \geq 1$
6   SomeAlg($A, \ell, h - t$)
7   SomeAlg($A, \ell + t, h$)
8   SomeAlg($A, \ell, h - t$)

Какво прави този алгоритъм, ако масивът е $A[1\,..\,n]$, $n \geq 1$ и началното викане е SomeAlg($A, 1, n$)?
Докажете това формално и прецизно.
Намерете сложността по време на SomeAlg.
Професор Дълбоков казва, че проверката на ред 1 е излишна и ред 1 може да се изтрие, като алгоритъмът остава коректен. Прав ли е професорът?

Решение: В алгоритмичния фолклор, този алгоритъм е известен като Stooge Sort. Той сортира, макар и по изключително неефикасен начин, откъдето идва и името му.

Ще докажем по индукция по n, че за всеки масив $A[1\,..\,n]$, SomeAlg($A, 1, n$) връща входния масив в сортиран вид. Доказателството не е (съвсем) тривиално поради начина, по който намалява размерът на входа на редове 6, 7 и 8.

Забележете, че ако $\ell = 1$ и $h = n$, на ред 5 променливата t получава стойност $\left\lfloor \frac{n-1+1}{3} \right\rfloor$, което е $\left\lfloor \frac{n}{3} \right\rfloor$.

На редове 6 и 8, рекурсивното викане е SomeAlg($A, 1, n - \left\lfloor \frac{n}{3} \right\rfloor$). Но за всяко $x \in \mathbb{R}$ е в сила $-\lfloor x \rfloor = \lceil -x \rceil$, така че $-\left\lfloor \frac{n}{3} \right\rfloor = \left\lceil -\frac{n}{3} \right\rceil$, така че

$$n - \left\lfloor \frac{n}{3} \right\rfloor = n + \left\lceil -\frac{n}{3} \right\rceil = \left\lceil n - \frac{n}{3} \right\rceil = \left\lceil \frac{2n}{3} \right\rceil$$

Тогава на редове 6 и 8 подмасивът, върху който става викането, е $A\left[1\,..\,\left\lceil \frac{2n}{3} \right\rceil\right]$. Тъй като $A\left[1\,..\,\left\lceil \frac{2n}{3} \right\rceil\right]$ има точно $\left\lceil \frac{2n}{3} \right\rceil$ елементи, рекурсивните викания на редове 6 и 8 са върху входове с размер $\left\lceil \frac{2n}{3} \right\rceil$.

Да разгледаме рекурсивното викане на ред 7. То е SomeAlg($A, 1 + \left\lfloor \frac{n}{3} \right\rfloor, n$). Подмасивът $A\left[1 + \left\lfloor \frac{n}{3} \right\rfloor\,..\,n\right]$ има точно

$$n - \left(1 + \left\lfloor \frac{n}{3} \right\rfloor\right) + 1 = n - \left\lfloor \frac{n}{3} \right\rfloor = n + \left\lceil -\frac{n}{3} \right\rceil = \left\lceil n - \frac{n}{3} \right\rceil = \left\lceil \frac{2n}{3} \right\rceil$$

елементи. Тогава и на ред 7, рекурсивното викане е върху вход с размер $\left\lceil \frac{2n}{3} \right\rceil$.

Заключаваме, че рекурсивните викания винаги са върху входове с размер $\left\lceil \frac{2n}{3} \right\rceil$.

Ключов факт, е че итераторът $n \mapsto \left\lceil \frac{2n}{3} \right\rceil$ има фиксирана точка 2 за всяко $n \in \mathbb{N}^+ \setminus \{1\}$.

На прост български, стартирайки от произволно $n \geq 2$, редицата

$$n \mapsto \left\lceil \frac{2n}{3} \right\rceil \mapsto \left\lceil \frac{2\left\lceil \frac{2n}{3} \right\rceil}{3} \right\rceil \mapsto \left\lceil \frac{2\left\lceil \frac{2\left\lceil \frac{2n}{3} \right\rceil}{3} \right\rceil}{3} \right\rceil \mapsto \cdots$$

неизбежно завършва с 2. Примерно,

$$\begin{aligned}
&3 \mapsto 2\\
&4 \mapsto 3 \mapsto 2\\
&5 \mapsto 4 \mapsto 3 \mapsto 2\\
&6 \mapsto 4 \mapsto 3 \mapsto 2\\
&7 \mapsto 5 \mapsto 4 \mapsto 3 \mapsto 2\\
&8 \mapsto 6 \mapsto 4 \mapsto 3 \mapsto 2\\
&9 \mapsto 6 \mapsto 4 \mapsto 3 \mapsto 2\\
&10 \mapsto 7 \mapsto 5 \mapsto 4 \mapsto 3 \mapsto 2
\end{aligned}$$

Това е от значение за верификацията на SOMEALG, понеже за всяко начално викане върху вход с ≥ 2 елемента, дъното на рекурсията е викане върху вход с два елемента. Ерго, условието на ред 1 винаги е истина при $n \geq 2$ и ред 1 спокойно може да бъде изтрит (За $n = 1$ условието не е истина, но въпреки това алгоритъмът работи коректно и без ред 1. Професор Дълбоков по изключение е прав!).

Лема 1 *Итераторът* $n \mapsto \left\lceil \frac{2n}{3} \right\rceil$ *има фиксирана точка* 2 *за всяко* $n \in \mathbb{N}^+ \setminus \{1\}$.

Доказателство: Със силна индукция по n. Базата е $n = 2$ и очевидно $2 \mapsto \left\lceil \frac{2n}{3} \right\rceil = 2$. Да допуснем, че твърдението е вярно за стойности на аргумента $2, 3, \ldots, n-1$, за някое $n - 1 \geq 2$; тоест, $n \geq 3$. Ще докажем твърдението за стойност на аргумента n. Щом $n \geq 3$, имаме

$$n > \left\lceil \frac{2n}{3} \right\rceil \geq 2$$

Тогава итераторът ще изобрази n в число, строго по-малко от n, но по-голямо или равно на 2. За това число, а именно $\left\lceil \frac{2n}{3} \right\rceil$, индуктивното предположение е в сила. Тогава очевидно твърдението за стойност на аргумента n е вярно. □

Теорема 2 *За всяко* $n \geq 1$, *за всеки* $A[1\,..\,n]$, SOMEALG$(A, 1, n)$ *сортира* $A[1\,..\,n]$.

Доказателство: Първо да разгледаме случая $n = 1$. Условието на ред 1 е лъжа и редове 2–8 не се изпълняват. Едноелементният масив A е който е тривиално сортиран в края на алгоритъма.

Нека $n \geq 2$. Доказателството е по индукция с база $n = 2$. Разглеждаме случая $n = 2$. Тогава $\boldsymbol{l} = 1$ и $\boldsymbol{h} = 2$. Забелязваме, че условието на ред 1 е истина, така че изпълнението отива на ред 2.

- Ако $A[1] > A[2]$, условието на ред 2 е истина и $A[1]$ бива разменен с $A[2]$ на ред 3.

- Ако $A[1] \leq A[2]$, условието на ред 2 е лъжа и ред 3 не се изпълнява.

Във всеки случай, масивът е сортиран и се състои от входните елементи, когато изпълнението е на ред 4. На ред 4, променливата t получава стойност $\left\lfloor\frac{2-1+1}{3}\right\rfloor = \left\lfloor\frac{2}{3}\right\rfloor = 0$. Тогава условието на ред 5 е лъжа и масивът в края на алгоритъма се състои от двата елемента на входа, но в сортиран вид. С което базата е доказана.

Доказателството е със силна индукция. Допускаме, че алгоритъмът работи коректно върху всички входове с размери $2, 3, \ldots, n-1$, за някакво $n-1 \geq 2$ (тоест, $n \geq 3$) и разглеждаме работата му върху произволен вход с размер n. Това означава, че $l = 1$ и $h = n$ в началото. Условието на ред 1 е истина и изпълнението отива на ред 2. Ако $A[1] > A[n]$, условието на ред 2 е истина и $A[1]$ бива разменен с $A[n]$ на ред 3, в противен случай A не бива променен.

> *Забележка: Размяната на $A[l]$ с $A[h]$ на ред 3 има значение за коректността само в базовия случай, в който масивът има два елемента. При масив с повече от два елемента, редове 2 и 3 спокойно може да бъдат прескочени.*

На ред 4, променливата t получава стойност $\left\lfloor\frac{n-1+1}{3}\right\rfloor = \left\lfloor\frac{n}{3}\right\rfloor$. Тъй като $n \geq 3$ в текущите допускания, вярно е, че $t \geq 1$. Поради това, условието на ред 5 е истина и редове 6, 7 и 8 се изпълняват.

Удобно е да се въведат следните означения:

$$X = A\left[1 \mathrel{..} \left\lceil\frac{n}{3}\right\rceil\right]$$
$$Y = A\left[\left\lceil\frac{n}{3}\right\rceil + 1 \mathrel{..} \left\lceil\frac{2n}{3}\right\rceil\right]$$
$$Z = A\left[\left\lceil\frac{2n}{3}\right\rceil + 1 \mathrel{..} n\right]$$

Това не са snapshots на подмасиви на A в някой фиксиран момент, а са кратки имена за подмасиви на текущия A, в който и момент да го разглеждаме. X, Y и Z представляват разбиване на A в смисъл, че всеки елемент на $A[1 \mathrel{..} n]$ е в точно един от тях. С "XY" ще означаваме $A\left[1 \mathrel{..} \left\lceil\frac{2n}{3}\right\rceil\right]$, а с "$YZ$" ще означаваме $A\left[\left\lceil\frac{n}{3}\right\rceil + 1 \mathrel{..} n\right]$. В някакъв смисъл, XY и YZ са конкатенации съответно на X и Y и на Y и Z. От ключово значение е, че викането на ред 6 е върху подмасива XY, викането на ред 7 е върху подмасива YZ и викането на ред 6 е отново върху подмасива XY.

В сила е

$$|X| = \left\lceil\frac{n}{3}\right\rceil$$
$$|Y| = \left\lceil\frac{2n}{3}\right\rceil - \left\lceil\frac{n}{3}\right\rceil$$
$$|Z| = n - \left\lceil\frac{2n}{3}\right\rceil$$

Случай 1: $n = 3k$ за някое $k \in \mathbb{N}^+$. Тогава $|X| = |Y| = |Z| = k$. На ред 6 има викане върху XY. Тъй като $|XY| = 2k$, вярно е, че $2 \leq |XY| < n$. Съгласно индуктивното предположение, това викане сортира подмасива XY на входа. Тогава при излизането от това викане, подмасивът Y съдържа k на брой най-големи елементи на входния XY.

Викането на ред 7 е върху YZ. Тъй като $|YZ| = 2k$, вярно е, че $2 \leq |YZ| < n$. Съгласно индуктивното предположение, това викане сортира подмасива YZ. Щом Y в началото съдържа k на брой максимални елементи на входния XY, след излизането от това викане, Z съдържа k на брой най-големи елементи на целия вход $A[1 \,..\, n]$, и то в сортиран вид. Тогава останалата част от масива, а именно XY, съдържа $2k$ на брой най-малки елементи на целия вход $A[1 \,..\, n]$.

Викането на ред 8 е върху XY. Както вече видяхме, то го сортира съгласно индуктивното предположение. Заключаваме, че след излизането от това викане, XY съдържа $2k$ на брой най-малки елементи на целия вход $A[1 \,..\, n]$, и то в сортиран вид.

Тогава в края на алгоритъма, масивът $A[1 \,..\, n]$ съдържа входните елементи в сортиран вид.

Случай 2: $n = 3k + 1$ за някое $k \in \mathbb{N}^+$. Тогава $|X| = k + 1$ и $|Y| = |Z| = k$. На ред 6 има викане върху XY. Тъй като $|XY| = 2k + 1$, вярно е, че $2 \leq |XY| < n$. Съгласно индуктивното предположение, това викане сортира подмасива XY на входа. Тогава при излизането от това викане, подмасивът Y съдържа k на брой най-големи елементи на входния XY.

Викането на ред 7 е върху YZ. Тъй като $|YZ| = 2k$, вярно е, че $2 \leq |YZ| < n$. Съгласно индуктивното предположение, това викане сортира подмасива YZ. Щом Y в началото съдържа k на брой максимални елементи на входния XY, след излизането от това викане, Z съдържа k на брой най-големи елементи на целия вход $A[1 \,..\, n]$, и то в сортиран вид. Тогава останалата част от масива, а именно XY, съдържа $2k+1$ на брой най-малки елементи на целия вход $A[1 \,..\, n]$.

Викането на ред 8 е върху XY. Както вече видяхме, то го сортира съгласно индуктивното предположение. Заключаваме, че след излизането от това викане, XY съдържа $2k+1$ на брой най-малки елементи на целия вход $A[1 \,..\, n]$, и то в сортиран вид.

Тогава в края на алгоритъма, масивът $A[1 \,..\, n]$ съдържа входните елементи в сортиран вид.

Случай 3: $n = 3k + 2$ за някое $k \in \mathbb{N}^+$. Тогава $|X| = |Y| = k + 1$ и $|Z| = k$. На ред 6 има викане върху XY. Тъй като $|XY| = 2k + 2$, вярно е, че $2 \leq |XY| < n$. Съгласно индуктивното предположение, това викане сортира подмасива XY на входа. Тогава при излизането от това викане, подмасивът Y съдържа $k + 1$ на брой най-големи елементи на входния XY.

Викането на ред 7 е върху YZ. Тъй като $|YZ| = 2k + 1$, вярно е, че $2 \leq |YZ| < n$. Съгласно индуктивното предположение, това викане сортира подмасива YZ. Щом Y в началото съдържа $k + 1$ на брой максимални елементи на входния XY, след излизането от това викане, Z съдържа k на брой най-големи елементи на целия вход $A[1\,..\,n]$, и то в сортиран вид. Тогава останалата част от масива, а именно XY, съдържа $2k+2$ на брой най-малки елементи на целия вход $A[1\,..\,n]$.

Викането на ред 8 е върху XY. Както вече видяхме, то го сортира съгласно индуктивното предположение. Заключаваме, че след излизането от това викане, XY съдържа $2k+2$ на брой най-малки елементи на целия вход $A[1\,..\,n]$, и то в сортиран вид.

Тогава в края на алгоритъма, масивът $A[1\,..\,n]$ съдържа входните елементи в сортиран вид. $\square$

Сложността по време на алгоритъма се изразява чрез следното рекурентно уравнение:

$$T(n) = 3T\left(\frac{2n}{3}\right) + 1 = 3T\left(\frac{n}{\frac{3}{2}}\right) + 1$$

Съгласно първия случай на МТ, решението е

$$T(n) \asymp n^{\log_{\frac{3}{2}} 3}$$

Тъй като $\log_{\frac{3}{2}} 3 \approx 2.709511292$, алгоритъмът е по-лош от квадратичен като сложност по време.

Както вече отбелязахме, професор Дълбоков е прав. Ред 1 може спокойно да бъде премахнат без коректността на алгоритъма да пострада. Причината е, че ако началният масив има размер $n \geq 2$, условието на ред 1 е истина и неизбежно се стига до рекурсивни викания върху масиви с размер точно 2. При $n = 2$ условието на ред 1 продължава да е истина. Дали условието на ред 2 е истина е без значение за терминирането на алгоритъма. Ако $n = 2$, изпълнено е $h = l + 1$, така че на ред 4, t получава стойност

$$\left\lfloor \frac{l + 1 - l + 1}{3} \right\rfloor = \left\lfloor \frac{2}{3} \right\rfloor = 0$$

така че условието на ред 5 е лъжа и текущото рекурсивно викане терминира. Ерго, размер 2 е спирачката на рекурсията, ако $n \geq 2$.

Ако $n = 1$, което е позволено по условие, $l = h$, условието на ред 2 е лъжа, ред 3 не се изпълнява, после t става 0 на ред 4, след което условието на ред 5 е лъжа и текущото рекурсивно викане терминира. Отново, условието на ред 1 е истина.

И така, условието на ред 1 е истина винаги, така че този ред може да се премахне – алгоритъмът ще остане коректен.