# МАТЕМАТИЧЕСКА ИНДУКЦИЯ

**Задача 1.** Докажете, че $2^{3n} - 7n - 1$ се дели на 49 за всяко цяло число $n \geq 0$.

**Решение:** Да означим израза с $f(n) = 2^{3n} - 7n - 1$.

Очевидно $f(0) = 0$ се дели на 49.

Нека допуснем, че $f(n)$ се дели на 49.

$$\begin{aligned} f(n+1) &= 2^{3(n+1)} - 7(n+1) - 1 \\ &= 2^{3n+3} - 7n - 7 - 1 \\ &= 2^3 2^{3n} - 7n - 8 \\ &= 8.2^{3n} - 8(7n+1) + 8(7n+1) - 7n - 8 \\ &= 8(2^{3n} - 7n - 1) + 56n + 8 - 7n - 8 \\ &= 8f(n) + 49n \end{aligned}$$

Представихме $f(n+1)$ като сума на $8f(n)$ и $49n$. Двете събираеми се делят на 49, следователно и $f(n+1)$ се дели на 49.

Получихме, че ако $f(n)$ се дели на 49, то $f(n+1)$ също се дели на 49.

По индукция следва, че $f(n) = 2^{3n} - 7n - 1$ се дели на 49 за всяко $n \geq 0$.

**Зад. 2.** Докажете, че $\forall n \geq 1 : \sum_{k=1}^{n} \frac{1}{\sqrt{k}} \geq \sqrt{n}$.

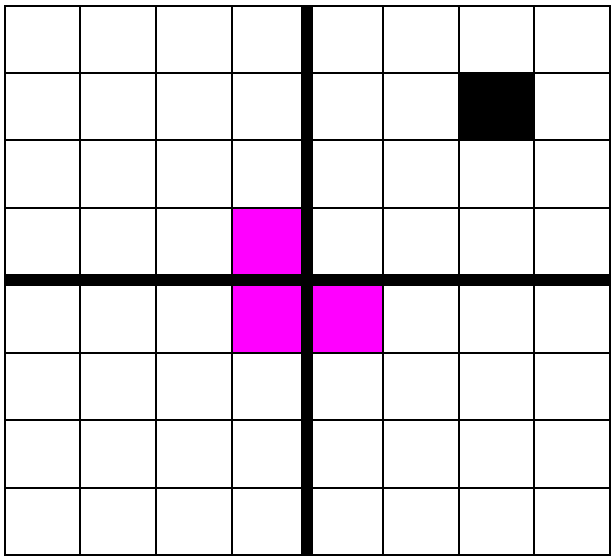
**Решение:** База: при n = 1 се получава вярното неравенство $1 \geq 1$.

Индуктивна стъпка:

$$\begin{aligned} \sum_{k=1}^{n+1} \frac{1}{\sqrt{k}} &= && \text{(от асоциативността на събирането)} \\ \left(\sum_{k=1}^{n} \frac{1}{\sqrt{k}}\right) + \frac{1}{\sqrt{n+1}} &\geq && \text{(от индуктивното предположение)} \\ \sqrt{n} + \frac{1}{\sqrt{n+1}} &= \frac{\sqrt{n}\sqrt{n+1}}{\sqrt{n+1}} + \frac{1}{\sqrt{n+1}} = \frac{\sqrt{n^2+n}+1}{\sqrt{n+1}} &&\geq \\ \frac{\sqrt{n^2}+1}{\sqrt{n+1}} &= \frac{n+1}{\sqrt{n+1}} = \sqrt{n+1}. \end{aligned}$$

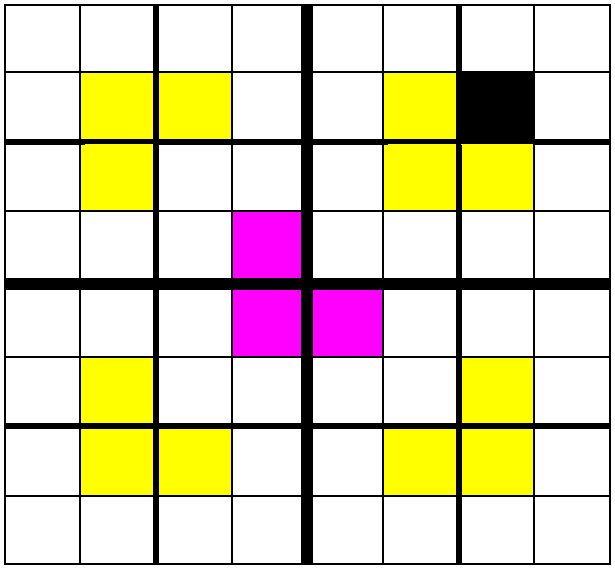
**Зад. 3.** Квадрат със страна $2^n$ е разделен на малки квадратчета със страна 1. Едно от тях е оцветено в черно, всички останали в бяло. На всеки ход можем да поставим тримино успоредно на страните на квадрата, стига то да попадне само върху бели квадратчета. (Тримино е плочка от три квадратчета под формата на буквата Г, тоест квадрат $2 \times 2$, на който липсва една клетка). Съставете алгоритъм за покриване на целия квадрат (без черното квадратче) с тримино, които не се застъпват и не стърчат извън квадрата.

Решение: Задачата се решава чрез *рекурсия*. Като метод за програмиране рекурсията означава една подпрограма да вика сама себе си (*пряка рекурсия*) или няколко подпрограми да се викат взаимно (*косвена рекурсия*). Обикновено при рекурсивното извикване се решава по-малък вариант на същата задача.



първа стъпка

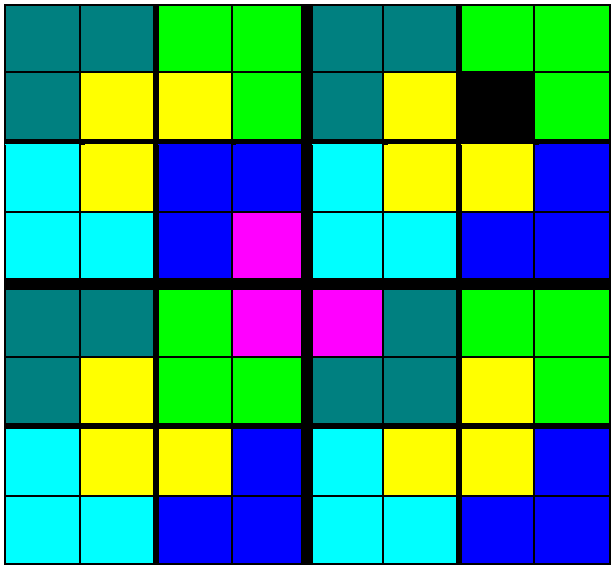
Конкретната задача се решава тъй: разделяме големия квадрат на четири по-малки квадрата със страни $2^{n-1}$, слагаме първото тримино в центъра така, че да закрива по едно квадратче от трите квадрата без липсващо поле. Сега всеки от четирите квадрата има по едно липсващо поле (черното или някое от трите розови полета). Остава да покрием с тримино всеки от четирите квадрата, а това е същата задача, но вече за $n - 1$ вместо $n$.



втора стъпка

На втората стъпка, прилагайки същата стратегия към получените четири квадрата, добавяме по едно тримино в центъра на всеки от тях (това са жълтите клетки). Делим всеки от четирите малки квадрата на четири още по-малки.

Продължаваме по този начин, докато стигнем до квадрати $2 \times 2$ (всеки с по едно липсващо поле). Другите три полета се запълват с по едно тримино.



последна (трета) стъпка

В конкретния пример задачата се решава за три стъпки. Фигурата вляво показва крайния отговор. Онези тримино, които са сложени на последната (третата) стъпка, са оцветени в оттенъци на зеления и синия цвят.

В изложеното решение използвахме начина на изразяване на програмистите: говорихме за алгоритъм и рекурсия. Решението обаче може да се формулира и на математически език: вместо “рекурсия” можем да казваме “индукция”; *дъното* на рекурсията (квадратът $2 \times 2$) съответства на *базата* на индукцията; свеждането на квадрат със страна $2^n$ до квадрат със страна $2^{n-1}$ представлява *индуктивната стъпка*.