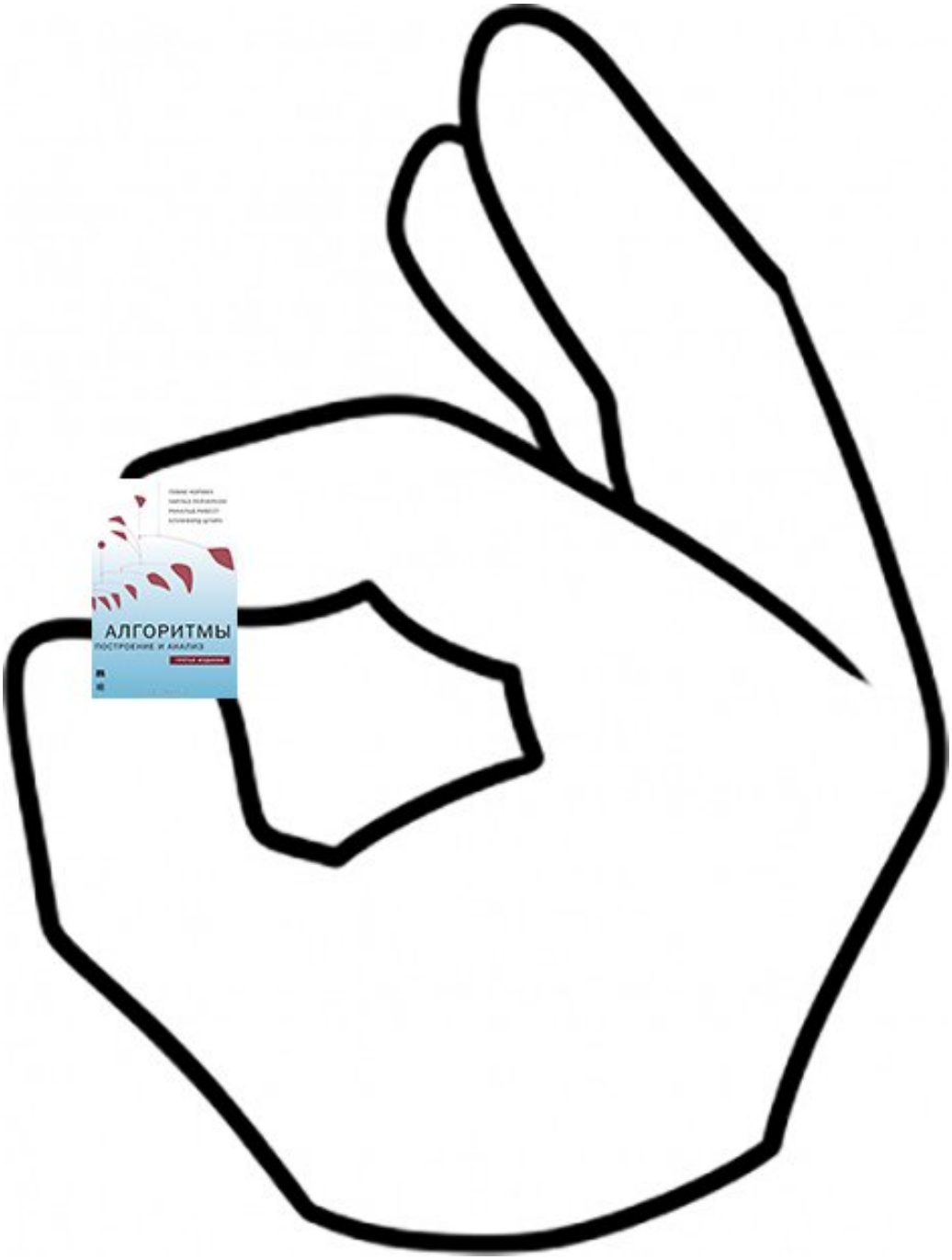
Алгоритмы и Структуры Данных

Мини-Кормен



Вадим Гринберг

Алексей Хачиянц

Содержание

1 Рекурсивные алгоритмы: задача о Ханойской башне. Оценка времени работы рекурсивного алгоритма при помощи рекуррентного соотношения. Доказательство оптимальности рекурсивного алгоритма.

1.1 Ханойские башни

Рассмотрим классическую задачу, предложенную Эдуардом Люка в 1883 году. Есть три стержня, при этом на первый стержень нанизано 8 дисков. Нужно перенести все диски на другой стержень, соблюдая два правила: диски можно двигать только по одному и нельзя класть диск большего радиуса на диск меньшего радиуса. Возможно ли это?

Придуманная профессором Люка легенда гласит, что в Великом храме города Бенарес, под собором, отмечающим середину мира, находится бронзовый диск, на котором укреплены 3 алмазных стержня, высотой в один локоть и толщиной с пчелу. Давным-давно, в самом начале времён, монахи этого монастыря провинились перед богом Брахмой. Разгневанный Брахма воздвиг три высоких стержня и на один из них возложил 64 диска, сделанных из чистого золота. Причем так, что каждый меньший диск лежит на большем.

Как только все 64 диска будут переложены со стержня, на который Брахма сложил их при создании мира, на другой стержень, башня вместе с храмом обратятся в пыль и под громовые раскаты погибнет мир.

Интуиция подсказывает, что это возможно. Но каков тогда алгоритм и сколько операций ему необходимо?

Чтобы переместить n дисков с первого стержня на второй, можно сначала переместить $n-1$ диск на третий стержень, перенести самый большой диск на второй стержень, а затем переместить $n-1$ диск с третьего стержня на второй. Такое рекурсивное решение представлено в Алгоритме 1.

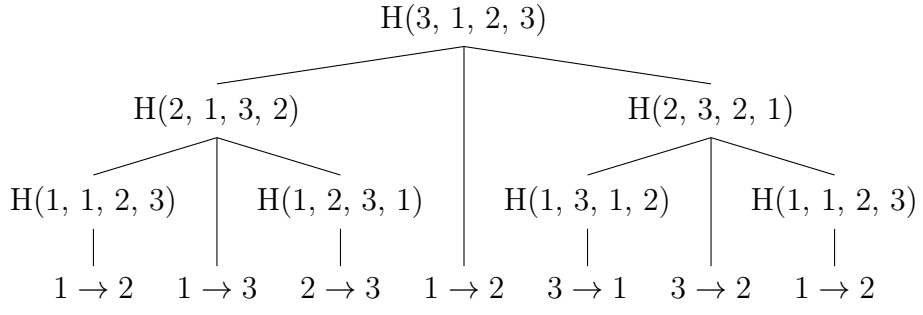
Алгоритм 1 Рекурсивный алгоритм решения задачи о Ханойской башне

Вход: Число дисков n, начальный i, конечный j и вспомогательный k-ый стержень.

Выход: Последовательность пар (x, y), соответствующих перемещению диска со стержня x на стержень y, приводящая к перемещению n верхних дисков со стержня i на стержень j.

```
1: function HANOI3(n, i, j, k)
2:     if n > 0 then
3:         HANOI3(n − 1, i, j, k)
4:         i → k
5:         HANOI3(n − 1, k, j, i)
```

Покажем с помощью дерева операций, как алгоритм работает для трёх дисков:



По сути, данный алгоритм обходит дерево в глубину, выполняя необходимые перестановки.

1.2 Оптимальность рекурсивного алгоритма и время его работы.

Докажем, что данный алгоритм является **оптимальным**, то есть нет алгоритма, который бы решал эту же задачу быстрее.

Пусть T_n — минимальное количество операций, за которое можно перенести n дисков. Сразу же заметим, что для переноса 0 дисков действий вообще не нужно. Тогда $T_0 = 0$.

Показанный ранее алгоритм показывает, что $T_n \leqslant 2T_{n-1} + 1$ для $n > 0$.

Возникает логичный вопрос: а можно ли быстрее? Увы, но нет. Рано или поздно придётся перенести самый широкий диск. Но для этого необходимо поставить $n - 1$ диск на один стержень. Тогда можно сделать вывод, что $T_n \geqslant 2T_{n-1} + 1$ для $n > 0$.

Отсюда получаем, что минимальное количество операций задаётся следующим *рекуррентным соотношением*:

$$T_n = \begin{cases} 2T_{n-1} + 1, & n > 0 \\ 0, & n = 0 \end{cases}$$

Прекрасно. Мы знаем, что алгоритм 1 наиболее оптимален и знаем рекуррентное соотношение, задающее количество операций. Но можно ли найти *замкнутую* формулу — такую, что она сразу даст нужное значение? Да, можно.

Теорема. $T_n = 2^n - 1$

Доказательство. Докажем это по индукции. База верна, так как $T_0 = 0 = 2^0 - 1$. Теперь пусть предположение верно для $n - 1$, то есть $T_{n-1} = 2^{n-1} - 1$. Тогда

$$T_n = 2T_{n-1} + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 2 + 1 = 2^n - 1 \qquad \square$$

Теперь немного изменим задачу.

1.3 Четыре стержня

Допустим, что у нас четыре стержня, два из которых — вспомогательные. Обозначим

$$n_m = \sum_{i=1}^{m} i = \frac{m(m+1)}{2}.$$

Для простоты предположим, что изначально на первом стержне находится n_m дисков.

Чтобы переместить n_m дисков с первого стержня на второй, можно сначала переместить n_{m-1} дисков на четвертый стержень, затем переместить оставшиеся m самых больших дисков на второй стержень, используя Алгоритм 1 и третий стержень в качестве вспомогательного, и наконец переместить n_{m-1} дисков с четвертого стержня на второй. Такое рекурсивное решение представлено в Алгоритме 2.

Алгоритм 2 Рекурсивный алгоритм решения задачи о Ханойской башне на 4-х стержнях

Вход: Число дисков $n_m = \frac{m(m+1)}{2}$, начальный i, конечный j и вспомогательные k и l стержни.

Выход: Последовательность пар (x, y), соответствующих перемещению диска со стержня x на стержень y, приводящая к перемещению n верхних дисков со стержня i на стержень j.

```
1: function HANOI4(n_m, i, j, k, l)
2:     if n > 0 then
3:         HANOI4(n − m, i, l, k, j)
4:         HANOI3(m, i, j, k)
5:         HANOI4(n − m, l, j, i, k)
```

Алгоритм 2 можно обобщить на случай произвольного числа дисков n, например, так: на верхнем уровне рекурсии выбрать n_{m-1}, ближайшее снизу к n, а вместо m в вызове алгоритма HANOI3 использовать $n - n_m$.

Обозначим за $g(n_m)$ число шагов, требующихся Алгоритму 2 для решения задачи с n_m дисками. Получаем следующее рекуррентное соотношение:

$$g(n_m) = \begin{cases} 0, & n_m = 0; \\ 2g(n_{m-1}) + 2^m - 1, & n > 0. \end{cases}$$

Докажем по индукции, что $g(n_m) = (m-1)2^m + 1$ (эту оценку можно получить, проанализировав дерево рекурсии).

Базис: $g(n_0) = 0 = -1 + 1 = (0-1) \cdot 2^0 + 1$.

Шаг: Предположим, что $g(n_{m-1}) = (m-2)2^{m-1} + 1$. Тогда

$$\begin{aligned} g(n_m) &= 2g(n_{m-1}) + 2^m - 1 = \\ &= 2((m-2)2^{m-1} + 1) + 2^m - 1 = \\ &= (m-2)2^m + 2 + 2^m - 1 = \\ &= (m-1)2^m + 1. \end{aligned}$$

Таким образом, число шагов асимптотически зависит от числа входных дисков n как $\Theta\left(\sqrt{n}\, 2^{\sqrt{2n}}\right)$. [1] Можно ли переместить диски быстрее — открытый вопрос.

[1] Смысл этого утверждения уточним на следующих лекциях.

2 O-, o-, Ω-, ω-, Θ- обозначения. Оценка сложности алгоритмов.

2.1 Асимптотические обозначения

Введем следующие обозначения:

$$\Theta(g(n)) = \{f(n) \mid \exists c_1 > 0, c_2 > 0 \exists n_0 : \forall n \geqslant n_0 \implies 0 \leqslant c_1 g(n) \leqslant f(n) \leqslant c_2 g(n)\},$$

Θ — *асимптотическое* $=$. Например, $2n = \Theta(n)$. По определению, $c_1 n \leqslant 2n \leqslant c_2 n$. Тогда $c_1 = 1, c_2 = 2$.

$$O(g(n)) = \{f(n) \mid \exists c_2 > 0 \exists n_0 : \forall n \geqslant n_0 \implies 0 \leqslant f(n) \leqslant c_2 g(n)\}$$

O — *асимптотическое* $\leqslant$. Например, по этому определению $n = O(n \log n)$, так как при достаточно больших n_0 следует, что $\log n_0 > 1$. Тогда $c_2 = 1$.

$$\Omega(g(n)) = \{f(n) \mid \exists c_1 > 0 \exists n_0 : \forall n \geqslant n_0 \implies 0 \leqslant c_1 g(n) \leqslant f(n)\}$$

Ω — *асимптотическое* $\geqslant$. Например, $n \log n = \Omega(n \log n)$ и $n \log n = \Omega(n)$. В обоих случаях подходит $c_1 = 1$.

$$o(g(n)) = \{f(n) \mid \forall c_2 > 0 \exists n_0 : \forall n \geqslant n_0 \implies 0 \leqslant f(n) \leqslant c_2 g(n)\}$$

o — *асимптотическое* $<$. Например, $n = o(n \log n)$. Покажем это. Пусть $n < c_2 n \log n \iff 1 < c_2 \log n \iff n > 2^{1/c_2}$. Тогда $n_0 = [2^{1/c_2} + 1]$

$$\omega(g(n)) = \{f(n) \mid \forall c_1 > 0 \exists n_0 : \forall n \geqslant n_0 \implies 0 \leqslant c_1 g(n) \leqslant f(n)\}$$

ω — *асимптотическое* $>$. Например, нельзя сказать, что $n \log n = \omega(n \log n)$. Но можно сказать, что $n \log n = \omega(n)$.

Заметим, что в логарифмах можно свободно менять основание: $\log_c n = \frac{\log_2 n}{\log_2 c}$. Именно поэтому не пишут основание логарифма.