Поздняков Сергей Николаевич

РЕКУРРЕНТНЫЕ ФОРМУЛЫ С ТОЧКИ ЗРЕНИЯ ИНФОРМАТИКИ И МАТЕМАТИКИ

ВВЕДЕНИЕ

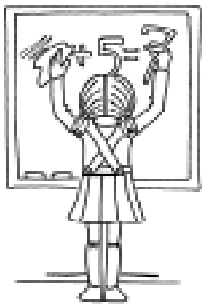
Когда от вычислений на бумаге люди стали переходить к вычислениям на компьютере, многим пришлось решить для себя новую психологическую проблему. Кажется, что, вычисляя на бумаге, мы используем неограниченную «память тетради», то есть мы всегда можем заглянуть на одну или несколько страниц назад. Но, чтобы писать программу для компьютера, надо сразу ограничить используемую память, зафиксировав ее для начальных, промежуточных и выходных данных. Напомним, что в первых вычислительных машинах память по современным понятиям была вообще ничтожна.

К сожалению, большинство алгоритмов действительно требует для своего выполнения неограниченной памяти. Например, вычисление цифр числа «пи» требует работы с уже найденными цифрами. Однако есть алгоритмы, которым бесконечная память не нужна. Например, сложение двух сколь угодно длинных натуральных чисел можно вести поразрядно, используя память только для сохранения величины переноса из разряда в разряд, двух текущих складываемых цифр и двух цифр результата.

В настоящем занятии мы рассмотрим класс задач, которые обладают таким свойством: вычисляя последующие значения интересующей нас величины, мы пользуемся только фиксированным числом последних значений. Последовательности с таким свойством называют рекуррентными.

ЧИСЛА ФИБОНАЧЧИ

Самой известной рекуррентной последовательностью является последовательность чисел Фибоначчи, которая начинается с двух единичек, а каждое следующее



...вычисляя последующие значения интересующей нас величины, мы пользуемся только фиксированным числом последних значений...

значение равно сумме двух предыдущих: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...

Конечно, без неограниченной памяти здесь тоже не обойтись, ведь длины самих чисел растут неограниченно, зато для вычислений достаточно всего два таких числа, а это значит, что мы можем оптимизировать использование памяти, по сравнению с хранением всех первых чисел Фибоначчи.

Задание 1.

Попробуйте оценить, насколько удалось уменьшить загрузку памяти компьютера удалением всех вычисленных чисел Фибоначчи, кроме двух последних.

...для вычислений достаточно всего два таких числа...

Уровень 1. В качестве оценки требуемой памяти возьмите сумму всех первых чисел Фибоначчи. Посчитайте суммы 2, 3, 4, 5, 6 первых чисел Фибоначчи. Можно ли эти суммы выразить через числа Фибоначчи? Проведите эксперимент. Сформулируйте гипотезу. Докажите ее по индукции.

Уровень 2. Возьмите теперь в качестве оценки требуемой памяти общее количество всех цифр в первых числах Фибоначчи. Напишите программу, которая по числу N первых чисел Фибоначчи выдает число цифр M в них. Например, если $N = 10$, то $M = 14$.

Уровень 3. Проведите исследование связи этих двух оценок требуемой памяти. Есть ли между ними какая-нибудь связь? Попробуйте найти приближенную зависимость, относительная погрешность которой уменьшается при росте N? Это означает, что, если $S = f(M)$ – точная, а $S = g(M)$ – приближенная зависимость, то значение $(f(M) - g(M))/f(M)$ становится сколь угодно малым при бесконечном увеличении N.

Рекуррентные последовательности задаются рекуррентными формулами или, как их еще называют, рекуррентными соотношениями или уравнениями. Например, последовательность чисел Фибоначчи задается такой формулой:

$$f_{n+1} = f_n + f_{n-1}, \qquad (1)$$

значения n берутся натуральными. Заметим, что в информатике счет обычно начинают с нуля, поэтому мы будем его тоже считать натуральным, однако, так как в формуле встречается индекс (номер числа Фибоначчи) $n - 1$, то подставлять в эту формулу можно только $n = 1, 2, 3...$

По этой формуле мы всегда найдем следующее число по двум предыдущим. Первые числа надо указать явно. Значения f_0 и f_1 называются начальными значениями нашего рекуррентного отношения.

Итак, равенства

$$f_{n+1} = f_n + f_{n-1}, \quad f_0 = 1, \quad f_1 = 1 \qquad (2)$$

полностью определяют последовательность чисел Фибоначчи.

Для рекуррентных соотношений представляет интерес нахождение их решений в так называемой явной или замкнутой форме. Это означает, что при подстановке в такую формулу любого значения n для получения результата надо будет сделать фиксированное количество операций, то есть число операций не зависит от n. Сравните с вычислением n-го числа Фибоначчи по рекуррентной формуле с начальных значений. Чем больше значение n, тем больше операций потребуется, то есть замкнутой формой исходную рекуррентную формулу назвать нельзя.

Для тех, кто не знает замкнутой формулы для чисел Фибоначчи, ее нахождение будет настоящим математическим открытием. Здесь можно приостановить чтение и попробовать найти эту формулу. Предупреждаем сразу, что будет это нелегко.

Мы попробуем найти это решение в виде показательной функции: $f_n = \lambda^n$. Подставляя в основное уравнение (1), получим: $\lambda^{n+1} = \lambda^n + \lambda^{n-1}$. Теперь сократим на λ^{n-1} и получим квадратное уравнение: $\lambda^2 = \lambda + 1$ или $\lambda^2 - \lambda - 1 = 0$. Это уравнение неожиданно имеет иррациональные корни:

$$\lambda_1 = \frac{1+\sqrt{5}}{2} \text{ и } \lambda_2 = \frac{1-\sqrt{5}}{2}.$$

Почему неожиданно? Потому что наша последовательность состоит из натуральных чисел, а в последовательностях

$$\left(\frac{1+\sqrt{5}}{2}\right)^n \text{ и } \left(\frac{1-\sqrt{5}}{2}\right)^n$$

целых чисел нет совсем!

Однако ситуация изменится, если эти последовательности сложить.

Решим вместе такую задачу.

Задача.

Написать программу, вычисляющую значения членов последовательности

$$s_n = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n$$

Решение.

Первое, что приходит в голову, использовать операции возведения в степень, встроенные в любой язык программирования, и написать программу из одной строч-

ки, вычисляющую значения s_n по введенному значению n.

Но тогда мы получим лишь приближенное значение s_n и никогда не узнаем, целое ли оно на самом деле! Попробуем найти рекуррентную формулу для s_n. Для этого умножим s_n на s_1 и попробуем выразить s_{n+1}:

$$s_n \cdot s_1 = \left(\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n\right)\cdot\left(\left(\frac{1+\sqrt{5}}{2}\right)+\left(\frac{1-\sqrt{5}}{2}\right)\right)=$$

$$=\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} + \left(\frac{1-\sqrt{5}}{2}\right)^{n+1} + \left(\frac{1+\sqrt{5}}{2}\right)^{n}\cdot\left(\frac{1-\sqrt{5}}{2}\right)+$$

$$+\left(\frac{1-\sqrt{5}}{2}\right)^{n}\cdot\left(\frac{1+\sqrt{5}}{2}\right)=\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} + \left(\frac{1-\sqrt{5}}{2}\right)^{n+1} +$$

$$+\left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\cdot(-1) + \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\cdot(-1) = s_{n+1} - s_{n-1}$$

Если заметить, что $s_1 = 1$, то получаем простую формулу:

$$s_n = s_{n+1} - s_{n-1} \text{ или } s_{n+1} = s_n + s_{n-1}.$$

Подстановкой в исходную формулу вычислим значение s_2, оно равно 3.

Видим, что все числа в последовательности целые. Последовательность s_n отличается от последовательности Фибоначчи, хотя определяется тем же рекуррентным соотношением.

Задание 2.

Уровень 1. Подберите такие A и B, чтобы получилась формула для чисел Фибоначчи

$$f_n = Aa_n + Bb_n,$$

где

$$a_n = \left(\frac{1+\sqrt{5}}{2}\right)^n \quad \text{и} \quad b_n = \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Задание 3.

Проведите исследование последовательности

$$u_n = \left(\frac{1+\sqrt{2}}{2}\right)^n + \left(\frac{1-\sqrt{2}}{2}\right)^n.$$

Уровень 1. Выведите рекуррентную формулу для u_n. Докажите, что все u_n рациональные числа, то есть задаются дробью $u_n = \frac{p_n}{q_n}$.

Уровень 2. Найдите рекуррентные соотношения для p_n и q_n.

Напишите программу, вычисляющую значения p_n и q_n

a) в пределах, допустимых типом ***integer***;

b) описываемых массивами цифр от 1 до N.

Если на вход этой программы подается значение n, то на выходе должны быть в случае

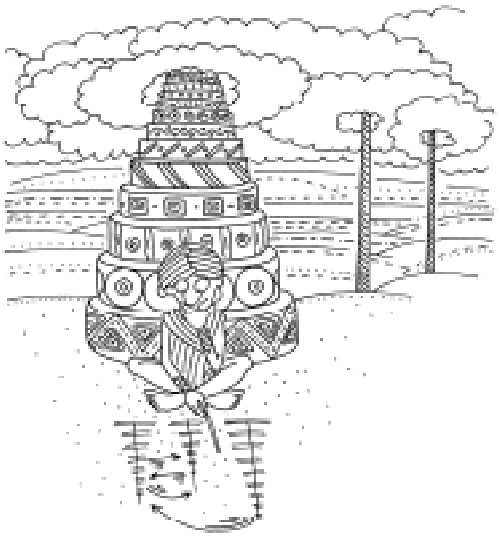
a) числа p_n и q_n;

b) массивы с цифрами чисел p_n и q_n.

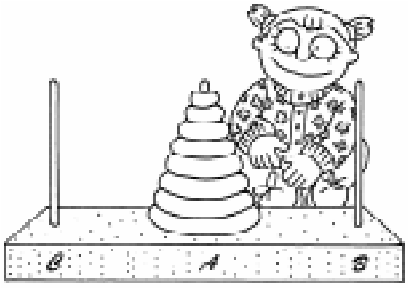
Замечание. Дробь $\frac{p_n}{q_n}$ не требуется делать несократимой.

Уровень 3. Степень $\left(\frac{1+\sqrt{2}}{2}\right)^n$ представляется в виде $\frac{A_n + B_n \cdot \sqrt{2}}{C_n}$. Найдите рекуррентные формулы для вычисления коэффициентов A_n, B_n, C_n.

ХАНОЙСКИЕ БАШНИ



Фабула новой задачи связана с легендой. При сотворении мира Всевышний, наряду с остальным, создал три алмазных шпиля. На один из них он водворил 64 диска из чистого золота: нижний самый большой, а каждый следующий меньше преды-



Более простой вариант с восемью дисками...

дущего. И повелел Всевышний своим служителям переместить диски на соседний по таким правилам: перемещать надо по одному диску за раз и при перемещении нельзя класть больший диск на меньший, для чего и нужен третий стержень. Как только жрецы закончат свою работу, наступит конец света.

Более простой вариант с восемью дисками математик Эдуард Люка предложил использовать как головоломку.

Попробуем применить к исследованию задачи метод рекуррентных соотношений.

Обозначим T_n – минимальное число перемещений башни из n дисков с одного стержня на другой по описанным правилам.

Чтобы переместить самый большой диск, надо сначала убрать с него все диски меньшего размера. Переместим их на промежуточный стержень. Для этого нужно сделать T_{n-1} перемещение. Затем положим большой диск на свое место – одно перемещение, а затем на него переместим всю башню из $n-1$ дисков с промежуточного стержня. Для этого понадобится еще T_{n-1} перемещение.

Таким образом, всего для перемещения башни из n дисков нужно сделать не менее $2T_{n-1}+1$ перемещений. Поскольку нас интересует минимальное число перемещений, получаем рекуррентное соотношение:

$$T_n = 2T_{n-1} + 1.$$

Начальное значение $T_1 = 1$ очевидно, так как башню из одного диска можно переместить за один ход.

Построим последовательно несколько значений T_n: 1, 3, 7, 15, 31, 63...

Тем, кто знаком с двоичной системой счисления, эти числа хорошо знакомы – их двоичная запись состоит из одних единиц. Иными словами, все эти числа на 1 меньше степеней двойки: 2 – 1, 4 – 1, 8 – 1, 16 – 1, 32 – 1, 64 – 1, ... Это можно доказать по индукции, а можно воспользоваться таким простым приемом: добавить к обеим частям рекуррентного соотношения по единичке и обозначить за $B_n = T_{n-1} + 1$: $T_n + 1 = 2T_{n-1} + 2 = 2(T_{n-1} + 1)$ или $B_n = 2B_{n-1}$. Поскольку $B_1 = 2T_1 = 2$, получаем, что $B_n = 2^n$, а значит $T_n = 2^n - 1$.

Задача 4.

Обозначим стержень, на котором изначально находятся диски – A, стержень, не котором диски должны в результате оказаться – B, а промежуточный стержень – C.

Изменим правила перемещения. Кроме тех, что были раньше, добавим запрет на прямое перемещение диска со стержня A на стержень B и обратно.

Уровень 1. Составьте рекуррентное соотношение для числа P_n – минимального числа перемещений башни из n дисков с одного стержня на другой по описанным правилам.

Уровень 2. Смоделируйте процесс оптимального перемещения дисков по этим правилам: если на вход подать значение t – количество перемещенных дисков (эту величину можно трактовать как время, требуемое для перемещений), то на выходе будут три массива – A, B и C, в которых находятся натуральные числа 1, 2, 3, 4, ..., n, обозначающие величины дисков. Массивы следует взять фиксированной длины 10, а оставшиеся неиспользованными элементы массива заполнить нулями. Увеличивать размер задачи, то есть число дисков не советуем, дабы не ждать окончания работы программы до конца света.

Уровень 3. Выведите замкнутую форму величины P_n.

Задание 5.

Изменим в предыдущей задаче ограничение следующим образом: переносить диск запрещается только напрямую со стер-

жня *A* на стержень *B* (а обратно уже разрешается).

Уровень 3. Проведите полное исследование этой задачи.

РАЗМЕН МОНЕТ

Не в каждой задаче удается легко построить замкнутую форму решения. Рекуррентные отношения позволяют описать более широкий класс задач. Рассмотрим задачу, которая раньше называлась размен монет, а сейчас (в связи с малым их использованием) назовем ее «размен купюр». У нас есть купюры достоинством в 5, 10, 50, 100 и 500 рублей в неограниченном количестве (!). Надо разменять купюру в 1000 рублей. Сколькими способами это можно сделать?

Значения всех купюр кратны 5, поэтому если взять 5 за единицу (ввести новую валюту, например), то задача займет меньше места: сколькими способами число 200 можно представить в виде слагаемых, значения которых равны 1, 2, 10, 20, 100 (разумеется, порядок слагаемых роли не играет).

Обозначим искомое число способов за R_n, тогда

$$R_n = R_{n-1} + R_{n-2} + R_{n-10} + R_{n-20} + R_{n-100}.$$

Объяснить это нетрудно. Раз порядок слагаемых нас не интересует, можно расположить их по возрастанию. Тогда наша сумма будет начинаться либо с 1, либо с 2, либо с 10, либо с 20, либо с 100. В каждом из этих случаев мы можем посчитать число способов разложить оставшуюся сумму. Поскольку других вариантов начала суммы нет, общее число комбинаций равно искомому.

Задание 6.

Уровень 1.

a) К этому рекуррентному соотношению надо добавить начальные условия, какие? Сколько их? Можно ли уменьшить их число за счет выписывания вспомогательных рекуррентных отношений?

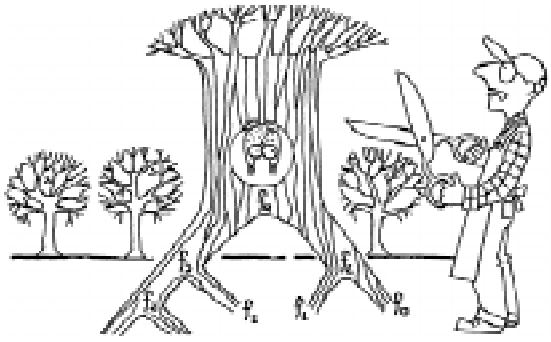
b) Выразите через R_n число разложений E_n, в состав которых обязательно входит единица.

Уровень 2. Составьте программу, вычисляющую количество представлений числа 200 в виде суммы слагаемых 1, 2, 10, 20, 100 (без учета порядка).

Уровень 3. Составьте программу для подсчета количества способов разложения числа N в сумму слагаемых 1, 2, 10, 20, 100, если количество слагаемых 1 ограничено числом c_1, количество слагаемых 2 ограничено числом c_2 и так далее – остальные границы соответственно c_{10}, c_{20}, c_{100}.

РЕКУРРЕНТНЫЕ ФОРМУЛЫ И РЕКУРСИЯ

На рекуррентное отношение, определяющее числа Фибоначчи, можно посмотреть иначе – как на их рекурсивное определение. В этом случае программа вычисления чисел Фибоначчи будет двигаться «в обратном направлении» – от определения к начальным значениям. Процесс этот обычно изображают в виде дерева. Напри-



Процесс этот обычно изображают в виде дерева.

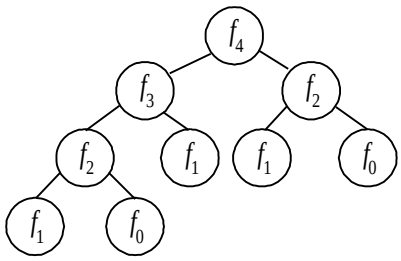


Рисунок 1.

мер, дерево вычислений f_4 изображено на рисунке 1.

Задание 7.

Сравним два способа вычисления чисел Фибоначчи. Для прямого – рекуррентного – способа нам надо хранить в памяти всего два текущих числа. А сколько памяти понадобится для обратного вычисления числа Фибоначчи по рекурсивному определению?

Уровень 1. Напишите рекуррентную формулу для вычисления числа

a) вершин дерева рекурсий числа Фибоначчи f_n;

b) вершин, помеченных числом f_1.

Выразите эти числа через числа Фибоначчи.

Мы видим, что использование рекурсии приводит к быстрому росту требуемой памяти. Для тех, кто нашел замкнутую формулу для чисел Фибоначчи, можем уточнить – «экспоненциальному росту».

...рассмотрим более безобидную версию задачи — детскую считалочку.

ЗАДАЧА ИОСИФА ФЛАВИЯ

Эта задача связана с легендой об известном историке первого века Иосифе Флавии. В этой легенде говорится о том, что он выжил и стал известным благодаря математической одаренности. В ходе иудейской войны он в составе отряда из 41 воина был загнан римлянами в пещеру. Предпочитая самоубийство плену, воины решили выстроиться в круг и последовательно убивать каждого третьего из живых до тех пор, пока не останется ни одного живого человека. Однако Иосиф, наряду с одним из своих единомышленников, счел подобный конец бессмысленным – он быстро вычислил спасительные места в круге, на которые поставил себя и своего товарища.

Оставим в стороне этическую сторону поступка Иосифа Флавия и рассмотрим более безобидную версию задачи – детскую считалочку.

Пусть по кругу стоят n человек, начинают считать с 1-го, и каждый k-ый выбывает. Кто останется водить?

Задание 8.

Уровень 1. Проведите несколько экспериментов для разных значений n и k ($1 \le n \le 10$, $2 \le k \le 5$) и результаты занесите в табличку. Напишите, какие закономерности Вы видите в этой табличке.

Уровень 2. Напишите программу, которая при вводе n и k выдает номер того, кто будет водить.

Уровень 3. Предположим, что в игре может участвовать от 3 до 20 человек, то есть n заранее неизвестно, поэтому просчитать игру заранее нельзя, значение k определяется в момент начала считалочки. Сформулируйте и обоснуйте ряд простых советов участнику игры, куда лучше встать, чтобы с меньшей вероятностью водить. Не беря ответственность за качество, приведем два примера таких советов: «если k – четное, то не становиться на четные места», «лучше всего стоять рядом с тем, с кого начинают считать». Для формулировки таких советов проведите компьютерный экспери-

мент, который статистически оценивает «качество» совета, то есть процент случаев, когда он «сработал», для различных значений n и k ($3 \le n \le 20$, $2 \le k \le 7$). Интересно было бы на вход программы подавать «совет» в некоторой формализованной форме, а программа должна выдавать его «качество».

Разберем теперь случай $k = 2$, то есть когда выбирают каждого второго.

На рисунке 2 изображен результат считалки для четного числа человек после первого прохода. В этом случае выбывают все игроки, стоящие на четных местах, а второй проход считалки опять начинается с первого игрока. Это все равно, как если бы мы начинали с вдвое меньшего числа людей, но номер каждого бы удваивался и уменьшался на 1.

Новые номера	1	2	3	4	...	n
Старые номера	1	3	5	7	...	$2n-1$

Если мы обозначим $J(n)$ – номер водящего после применения считалки к n игрокам, то получим рекуррентную формулу:

$$J(n) = 2J(n/2) - 1$$

Посмотрим теперь, что будет в случае нечетного числа игроков. После выбывания всех игроков с четными номерами выбывает первый игрок и счет для нового прохода начинается уже с номера 3 (рисунок 3).

В этом случае мы опять можем ввести новые номера и свести задачу к более простой. Но закон, связывающий новые номера со старыми, будет иной.

Новые номера	1	2	3	4	...	n
Старые номера	3	5	7	9	...	$2n+1$

Соответственно, другой будет и рекуррентная формула для участника, остающегося водить:

$$J(n) = 2J((n-1)/2) + 1$$

Заметим, что $(n-1)/2$ в этой формуле показывает, сколько игроков остается на второй круг.

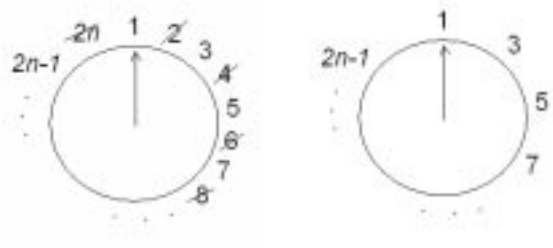


Рисунок 2.

Итак, получается такая рекуррентная зависимость:

$J(n) = 1$,

$J(n) = 2J(n/2) - 1$, если n четное,

$J(n) = 2J((n-1)/2) + 1$, если n нечетное.

Задание 9.

Уровень 1. Посчитайте по этой формуле $J(1000)$.

Уровень 2. Напишите программу, которая по вводу n выдает $J(n)$. Здесь полагаем, что n описывается типом ***integer***.

Посмотрим, что нам дает рекуррентная формула. При выполнении задания 8 (уровень 2) нам нужно было хранить номера всех n игроков (если, конечно, Вы уже не вывели рекуррентную формулу сами!). Таким образом, объем памяти, требуемый для решения задачи, менялся пропорционально «размеру задачи», как для таких целей именуют число n. Теперь мы можем организовать рекурсивное вычисление $J(n)$. Интуитивно ясно, что для этого понадобится гораздо меньше памяти, ведь за одно вычисление мы будем уменьшать n не менее чем вдвое.

Уровень 3. Напишите точную формулу для объема памяти, необходимого для вычисления числа $J(n)$ рекурсивно.

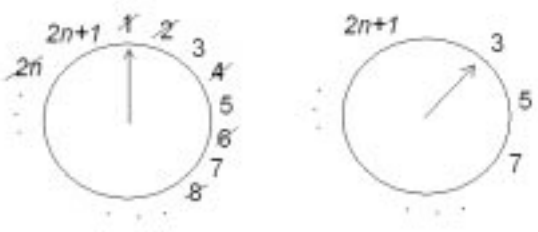


Рисунок 3.

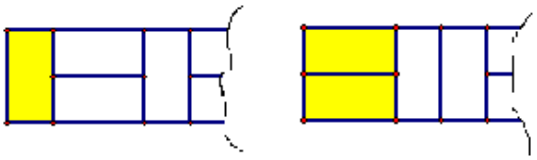


Рисунок 4.

А нельзя ли вообще обойтись без дополнительной памяти для вычислений $J(n)$, то есть ограничиться только памятью для хранения самого числа n?

Оказывается, можно!

Теорема. Если $n = 2^m + k$, где m, k – целые, причем $m \geq 0$, $0 \leq k < 2^m$, то $J(n) = 2n + 1$.

Задание 10.

Уровень 1. Вычислите по этой формуле $J(1000000)$.

Уровень 2. Напишите программу, которая при вводе массива с цифрами числа n в двоичной системе на выходе дает массив с двоичными цифрами числа $J(n)$.

Уровень 3. Исследуйте числа $J(J(...J(n))...)$. Обратите внимание, что с некоторого момента дальнейшее вычисление функции J не приводит к изменению результата. Объясните этот эффект. Найдите способ, как по двоичной записи числа n определить результат «бесконечного» применения функции J.

РЕКУРРЕНТНЫЕ СООТНОШЕНИЯ ОТ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ

ЗАДАЧА ПРО ДОМИНО

Задача.

Имеется полоска шириной 2 клеточки и длиной n клеточек. Мы хотим замостить ее костяшками домино размером 2×1. Костяшки можно класть вертикально и горизонтально. Сколькими способами это можно сделать, если ровно k из костяшек стоят вертикально, а остальные горизонтально (рисунок 4)?

Решение.

Обозначим число таких способов $N(n; k)$.

Если первая костяшка стоит вертикально, то остается замостить полоску размером $2 \times (n-1)$, что можно сделать $N(n-1, k-1)$ способами. Если же первая костяшка лежит горизонтально, то под ней (или на ней) должна обязательно быть еще одна такая же. Таким образом, остается замостить полоску размера $2 \times (n-2)$, что можно сделать $N(n-2, k)$ способами.

Получаем рекуррентное соотношение $N(n; k) = N(n-1, k-1) + N(n-2, k)$.

Оно содержит две переменные n и k, поэтому для вычисления $N(n, k)$ нужно знать все предыдущие значения, которых $n \cdot (k-1) + k \cdot (n-1)$. Для вычислений можно использовать массив с двумя переменными.

Задание 11.

Уровень 1. Для вычислений $N(n, k)$ нужны начальные значения. Сколько таких значений нужно? Напишите замкнутые формулы для вычисления начальных значений.

Уровень 2. Напишите программу для вычисления $N(n, k)$ по заданным значениям n и k, используя для хранения промежуточных данных массив от двух переменных. Можете ли Вы предложить более удобную структуру данных, чем массив?

Рассмотрим теперь пример, когда рекуррентная формула зависит от очень большого числа переменных: настолько большого, что мы даже не будем пытаться их явно перечислять.

РАСКРАСКА ГРАФА

Задача.

Сколькими способами можно раскрасить вершины заданного графа так, чтобы вершины, соединенные ребром, имели разный цвет? Всего разрешается использовать k различных цветов (рисунок 5).

Написать явную рекуррентную формулу мы не можем, поскольку не знаем каков граф. Однако мы можем разбить все раскраски на два класса и подсчитать число раскрасок в каждом из классов, а потом сложить. Найдем две вершины графа, не соединенные ребром, тогда возможны два случая раскраски: либо эти вершины окрашены в один цвет, либо в разные цвета. В первом случае эти вершины можно «склеить» в одну и свести задачу к подсчету числа раскрасок графа с меньшим на единицу числом вершин. В другом случае, наоборот, можно добавить ребро, соединяющее эти вершины. При этом раскраска останется правильной, так как мы соединяем разноокрашенные вершины.

Условно эту рекуррентную формулу можно изобразить так:

На первый взгляд может показаться, что, увеличивая число ребер, мы усложняем задачу, но это не так. Рассмотрим, например, крайний случай, когда все n вершин графа соединены (такой граф называется полным и обозначается K_n). В этом случае подсчет числа раскрасок очень прост: окраску начнем с любой вершины. Ее мы можем раскрасить в любой из k цветов. Возьмем любую другую вершину. Она соединена с первой ребром, поэтому ее можно окрасить в любой из $k - 1$ цветов. Всего комбинаций раскраски будет $k(k - 1)$. Далее берем третью вершину, она соединена с двумя предыдущими, поэтому для нее можно выбрать любой цвет, кроме двух уже выбранных. Всего будет $k(k - 1)(k - 2)$ комбинаций. Продолжая подсчет дальше, получим теорему.

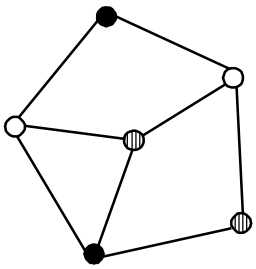


Рисунок 5.

Теорема. Число раскрасок полного графа с n вершинами k красками равно

$$k! = k \cdot (k - 1) \cdot (k - 2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1.$$

На рисунке 6 показан пример $n = 3$, $k = 3$.

Таким образом, наш метод ведет либо к уменьшению вершин графа, то есть к более простым графам, либо к полным графам, число раскрасок которых мы знаем.

Применяя нашу рекуррентную формулу, мы всегда можем свести задачу к раскраске полного графа.

Сколькими способами можно раскрасить вершины заданного графа...

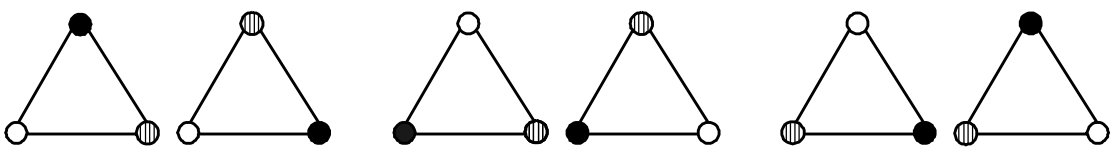


Рисунок 6.

Пример.

Сколькими способами можно раскрасить вершины четырехугольника в 4 цвета?

На рисунке 7 показан в символической форме ход решения.

Справа получились четыре полных графа: один с четырьмя вершинами, два – с тремя, один – с двумя.

Таким образом, ответ: 4×3×2×1 + + 3×2×1 + 3×2×1 + 2×1 = 24 + 6 + 6 + 2 = = 38

Задание 12.

Уровень 1. Посчитайте, сколькими способами можно раскрасить вершины пятиугольника в

a) два цвета,
b) три цвета,
c) четыре цвета,
d) пять цветов,
e) шесть цветов.

Уровень 2. Пусть граф имеет вершины, занумерованные числами 1, 2, 3, ..., n. Ребра графа заданы массивом $E(i, j)$; если вершины i и j соединены ребром, то $E(i, j)$ равно 1, иначе 0. Заметим, что мы храним избыточную информацию, так как $E(i, j) = E(j, i)$.

Напишите программу, которая считает число раскрасок этого графа k красками.

Рисунок 7.

Литература

1. *Р. Грэхем, Д. Кнут, О. Паташник.* Конкретная математика. Основания информатики: Пер. с англ. М.: Мир, 1998.

2. *F M Dong, K M Koh, K L Teo.* Chromatic polynomials and chromaticity of graphs. World Scientific Publishing, 2005.

Поздняков Сергей Николаевич,
доктор педагогических наук,
профессор кафедры ВМ-2 СПбГЭТУ.