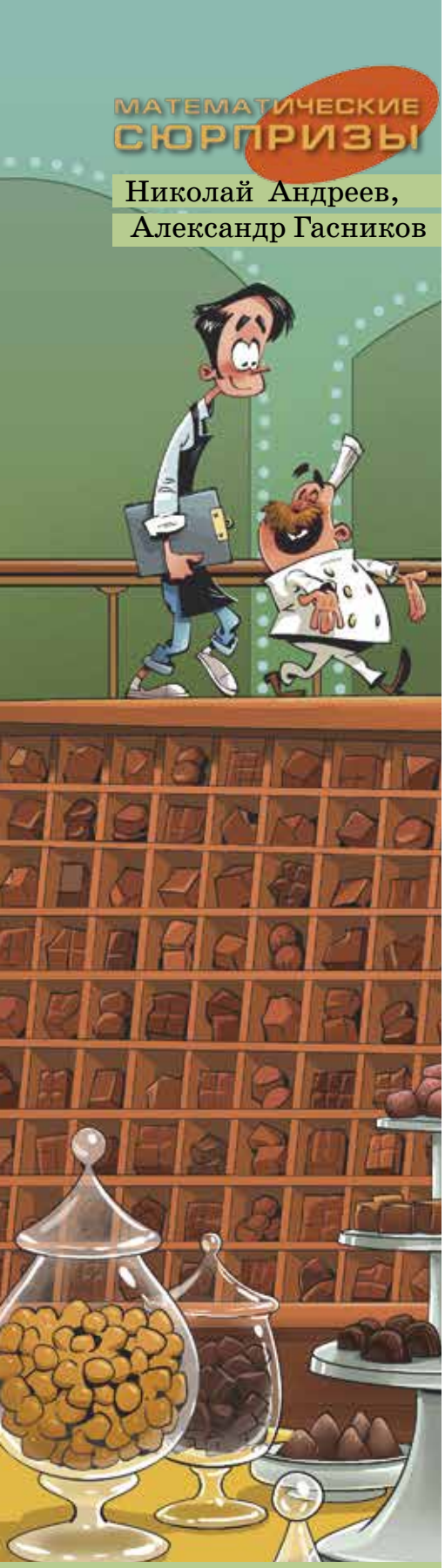
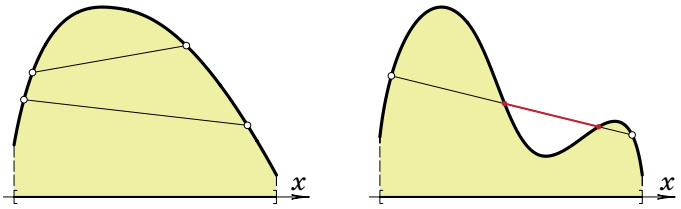


ПОИСК САМОЙ ВКУСНОЙ ШОКОЛАДКИ

Шоколад делается на основе масла какао, которое получается из какао-бобов – семян шоколадного дерева. Чтобы шоколад был вкусным, в масло примешивают множество добавок. Делается много вариантов, в которых добавки смешиваются в различных пропорциях, а дегустаторы выбирают вариант, по их мнению, самый вкусный.

Рассмотрим самую простую задачу, когда все ингредиенты уже зафиксированы, и мы должны определиться только с одной добавкой – например, сколько класть сахара. Если сахара мало, шоколад будет слишком горьким, а если слишком много – шоколад будет приторным. Выберем одного эксперта-дегустатора и введём функцию «вкусноты»: на вход подаётся количество добавленного сахара, и чем шоколадка вкуснее, тем больше значение функции вкусноты.

Будем рассматривать нашу функцию над отрезком, каждая точка которого показывает, сколько сахара положили в шоколад, а края отвечают каким-то разумным значениям количества сахара. Сделаем естественное допущение: будем считать, что наша функция вкусноты *строго выпукла вверх* – если соединить две любые точки на графике функции, то отрезок будет всегда лежать под графиком функции (выходя на график только своими концами). То есть между отрезком и графиком функции – выпуклая область, как на левой картинке, а не на правой.

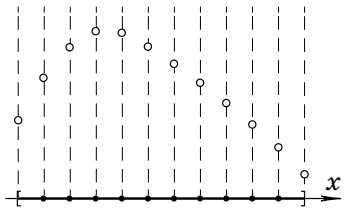


Это допущение основано на реальном эффекте насыщения – первые увеличения сахара оказывают большее влияние на вкус, чем следующие за ними. Можно считать, что поведение нашей функции на отрезке таково: функция вкусноты при увеличении количества сахара возрастает до какого-то момента (пока сахара слишком мало), в какой-то точке дости-

гает единственного на отрезке максимума, а потом убывает (когда сахара чересчур много).

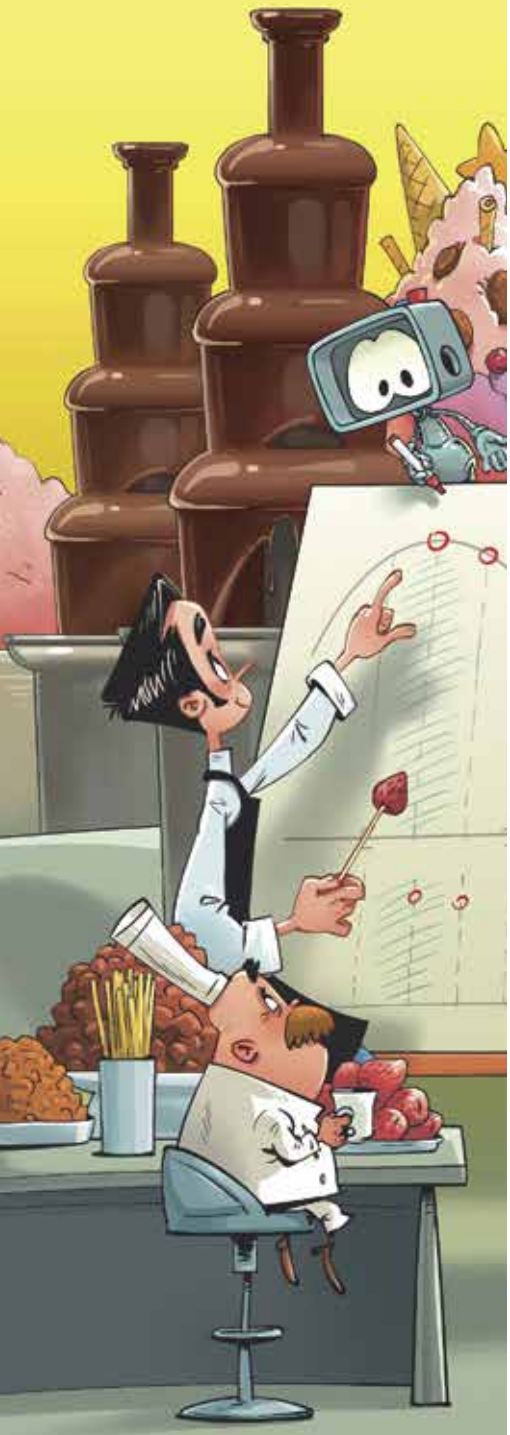
Мы не знаем, как в точности выглядит функция вкусноты, но хотим найти ту единственную точку, в которой она принимает максимальное значение: найти количество сахара, при котором шоколадка самая вкусная из возможных. По какому алгоритму нам действовать?

Первое, что приходит на ум: можно разбить отрезок на много-много шажков, пусть одинаковых; сделать для каждой точки шоколадку с соответствующим количеством сахара и заставить дегустатора все их попробовать. В качестве первой рассмотрим задачу, когда дегустатор умеет, попробовав шоколадку, указать для неё значение функции «вкусно-та». То есть мы умеем вычислять значение функции для любой точки, и требуется найти её максимум на заданном отрезке. Дегустатор, испытав тысячи шоколадок (а в реальных задачах числа оказываются гораздо больше), может выбрать самое большое значение из названных им (по сути, по точкам построить график функции вкусноты). Помня, на какой шоколадке достигался максимум, мы получим рецепт, близкий к наилучшему.

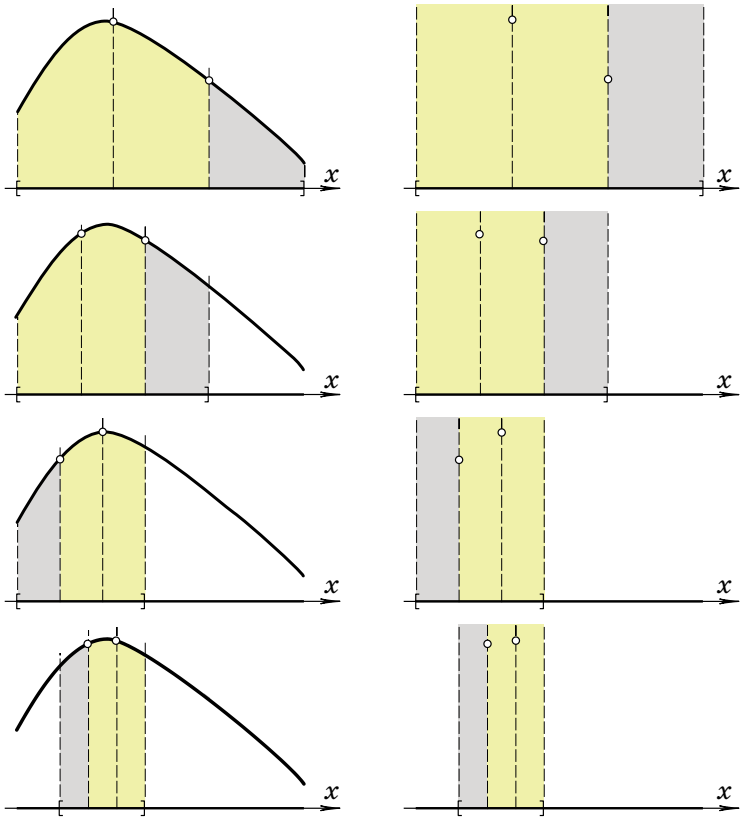


Но согласитесь, это очень странное требование к дегустатору, чтобы он по шоколадке назвал точное число – значения функции вкусноты. Чаще всего нормальный человек может сравнить два данных ему образца, сказав, что один вкуснее другого. И это вторая задача, которую мы попробуем решить. В этой постановке у нас есть неизвестная функция (вкусноты) и дегустатор, который по конкретным двум точкам на отрезке может сказать, что больше – значение функции в первой точке или второй. Требуется найти максимум этой функции на отрезке (точку, в которой он достигается).

Один из способов, используемый в информатике для решения обеих задач, – троичный поиск. Разделим отрезок двумя точками на три равные части и сравним значения в этих точках. Максимум функ-



ции (указанного типа) лежит в тех двух областях, которые граничат по точке с большим значением. Поэтому другая треть исходного отрезка – от точки, где значение меньше, до ближайшего к нему края – откидывается. (Если значения в двух точках совпали, то можно откинуть любой из крайних отрезков.) Новый отрезок на треть меньше предыдущего, и описанный шаг алгоритма повторяется применительно уже к нему. Тем самым, на каждом шаге отрезок, на котором находится искомый максимум функции, во-первых, лежит внутри предыдущих, во-вторых, с каждой итерацией уменьшается – сокращается на треть длины отрезка, рассматриваемого на предыдущем шаге, и для каждого шага приходится вычислять значение функции в двух точках, чтобы сравнить их.



В левом столбце первые четыре шага алгоритма показаны для случая первой задачи, когда мы умеем вычислять функцию. Но и в случае второй задачи, когда вычислять функцию не умеем, а умеем только сравнивать значения в двух точках, алгоритм, как показано в правом столбце, работает так же.

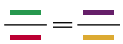
В реальных задачах очень часто точку максимума достаточно локализовать с какой-то точностью – указать маленький отрезок, внутри которого она находится. Действительно, если локализовать наилучшее количество сахара с точностью до отрезка в $\frac{1}{1000}$ от исходного, вряд ли даже профессиональный дегустатор почувствует разницу – дальнейшие уточнения не имеют практического смысла.

Чтобы локализовать точку максимума на отрезке в $\frac{1}{1000}$ от исходного в первой задаче (когда мы умеем вычислять функцию), необходимо было бы съесть тысячу шоколадок! А к нашей второй задаче алгоритм поточечного построения графика и вовсе не применим. Чтобы локализовать точку максимума с такой же точностью рассмотренным алгоритмом троичного поиска, придётся сделать всего 16 шагов: чтобы $\left(\frac{2}{3}\right)^n$ стало меньше $\frac{1}{1000}$. И, соответственно, попробовать $2 \cdot 16 = 32$ шоколадки. Всего 32 шоколадки, а не тысячу! Причём этот алгоритм работает и в первой задаче, и во второй. А можно ли с такой же точностью локализовать точку максимума, проявив заботу о дегустаторе и потратив ещё меньше денег и времени на эксперименты – за ещё меньшее число шоколадок?

Один из способов, который позволяет это сделать, – метод золотого сечения. Напомним читателю, что *золотым сечением* называется число $\varphi = \frac{1+\sqrt{5}}{2}$, которое удовлетворяет уравнению $\varphi^2 - \varphi - 1 = 0$. У этого числа много интересных свойств, и оно уже не раз встречалось на страницах нашего журнала.

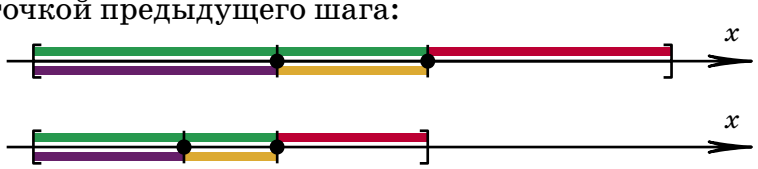
Метод золотого сечения улучшает алгоритм троичного поиска за счёт такого разбиения отрезка двумя точками на три (уже не равные) части, чтобы одна из точек совпадала с использовавшейся на предыдущем шаге. Тогда на очередном шаге информацию в этой точке можно брать из предыдущего шага.

Две точки внутри отрезка будем выбирать симметрично относительно середины отрезка

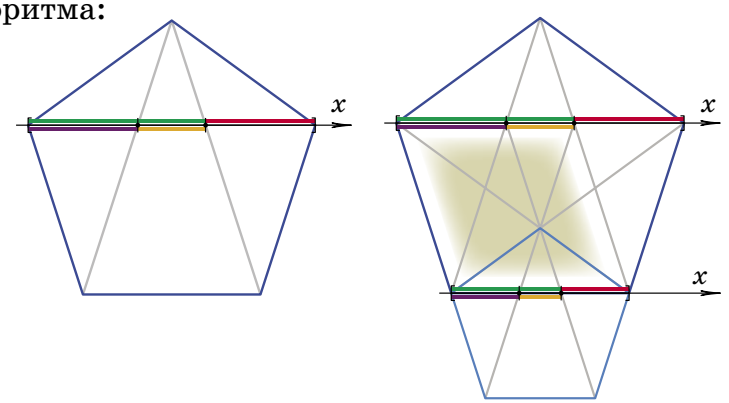
и так, чтобы выполнялось соотношение 

(а красный отрезок равен фиолетовому из-за симметричного выбора точек). При этом окажется, что оба эти отношения равны φ, откуда и берёт название описываемый метод. При таком разбиении это же соотношение выполняется и для другой стороны отрезка:

Так как для золотого сечения $\frac{1}{\varphi} = \varphi - 1$, когда мы поделим отрезок следующего шага двумя точками в той же указанной пропорции, одна из них совпадёт с точкой предыдущего шага:



Любители геометрии наверняка оценят такое представление используемого деления отрезка и шага алгоритма:



В случае, когда мы ищем максимум функции, которую умеем вычислять, на первом шаге, конечно, придётся посчитать её значение в двух точках внутри отрезка. Но на каждой следующей итерации в совпадающей для двух шагов точке используется значение функции, посчитанное на предыдущем шаге, и считать надо только одно значение.

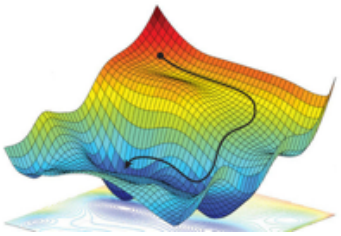
Когда функция неизвестна, алгоритм тоже работает. Помня вкусовые ощущения предыдущего шага, дегустатор, пробуя одну шоколадку, сравнивает её вкусность с тем, что уже было.

Итак, преимущество метода золотого сечения над алгоритмом троичного поиска основано на двух идеях. Во-первых, на каждом шаге надо пробовать одну шоколадку, а не две! Во-вторых, на каждом шаге от-

резок уменьшается не на $\frac{1}{3} = 0{,}3333...$ от длины предыдущего, а больше – на $\frac{1}{\varphi+1} = 0{,}3819...$ от длины, а значит, для достижения нужной точности надо будет сделать чуть меньше шагов. Выигрыш, по сравнению с троичным поиском, получается больше чем в 2 раза!

Это только в детстве кажется, что много шоколадок – это хорошо. Но много шоколадок съесть невозможно. Да и до́роги эксперименты с миллионами и миллиардами (а именно такие масштабы встречаются в реальных задачах) шоколадок. Количество съедаемых шоколадок при решении задачи – а на научном языке количество шагов алгоритма – необходимо минимизировать, делать как можно меньше. Раздел математики, который занимается вопросами эффективности алгоритмов, – это *теория сложности*, см., например, book.etudes.ru/articles/complexity/ в интернете.

А исходная задача – численного поиска максимума или минимума какой-то функции – относится к области, называемой *оптимизацией*. Мы с вами рассматривали простейшую постановку – с одним параметром: наша функция «жила» над отрезком, да ещё и обладала свойством выпуклости. Если параметров два, то функция задаёт поверхность над участком плоскости: мы попадаем с вами в холмистый, а иногда и горный, рельеф. Бродя по нему, мы должны найти самую низкую точку в заданном районе. Причём некоторые задачи надо решать, имея карту, а в некоторых задачах – даже не представляя карты местности, а только видя ближайший рельеф: по сути, бродя «на ощупь» в густом тумане! Как нужно строить маршрут поиска самой низкой точки, чтобы он был и алгоритмичным, и покороче? В современных реальных задачах количество параметров исчисляется миллиардами, поверхности строятся в многомерных пространствах и даже при наличии самых современных компьютеров необходимо находить математические алгоритмы, прокладывающие путь к точке минимума или максимума не за вечность, а за разумное время, как можно меньшее.



Художник Мария Усеинова