Содержание

Глава 1

Мышление будущего

Вычислительное мышление — это важный навык, который в информатике осваивают и используют для решения различных задач. Вычислительное мышление настолько важно, что во многих странах его преподают в средних школах. Но в чем же оно состоит? Как оно изменило практически все сферы нашей деятельности? И как использовать его для отдыха и развлечений?

Что с ним делать?

Представьте, что вы ученый и пытаетесь понять поведение птиц, которые ищут корм на земле. Одни кормятся, а другие смотрят в небо, чтобы вовремя увидеть хищника. Как они распределяют роли? Другие ученые тратят время на наблюдения за птицами, но вы идете дальше. Вы придумываете алгоритм — последовательность шагов, — которому должны следовать птицы, чтобы распределить роли. Потом вы создаете компьютерную модель и симулируете сценарии на основе гипотезы, что каждая птица смотрит на своих соседей. Это не только совпадает с результатами ваших наблюдений, но еще и позволяет делать прогнозы, поддающиеся проверке.

Возможно, вы фокусник и у вас появилась идея нового фокуса, основанного на математических свойствах чисел. Вы продумали этапы его представления, однако не уверены, что фокус получится при любых условиях. Вместо того чтобы опробовать

его, вы начинаете рассуждать логически и приходите к выводу, что в определенной ситуации все может пойти не так. Внеся некоторые изменения в демонстрацию фокуса, вы гарантируете, что эта ситуация никогда не возникнет.

Вы учитесь в школе, и учитель объясняет, как функционирует мозг. На доске нарисован нейрон и помечены его части, все это нужно запомнить. За вечер вы пишете программу, которая ведет себя как нейрон. Объединив несколько «нейронов» вместе, вы видите, каким образом состоящая из них группа может выполнять какие-то действия. На следующий день с помощью этой программы вы объясняете друзьям, как все работает.

Или, возможно, вы — доктор, которого расстраивают ошибки медицинского персонала при пользовании определенными приборами. Администрация винит сотрудников — одну медсестру только что уволили за такую ошибку. Вы осознаете, что проблема — в устройстве прибора. Занятому человеку легко сделать промах. Вы обращаетесь к производителям и показываете, что небольшие изменения в конструкции гарантируют, что проблема больше никогда не возникнет.

Или, например, вы — учитель и у вас огромная кипа проверенных работ, сложенных в произвольном порядке. Их надо рассортировать, чтобы на родительском собрании можно было быстро найти ту, о которой вы хотите поговорить. Но ничего страшного. Вы знаете, как быстро разложить их по порядку.

Еще один пример. На каникулах вы работаете в кофейне и замечаете, что у вас всегда длинная очередь и это огорчает клиентов. Вы говорите боссу, что сотрудник за кассой проводит много времени в бездействии и отчасти в этом заключается причина задержки. Если вся команда будет работать вместе, то очередь пойдет гораздо быстрее.

Может быть, у вас есть масса идей для игр, в которые вы с удовольствием играли бы с друзьями. Но, в отличие от остальных, вы не просто рассказываете о своих блестящих идеях, а пишете программы на их основе — и через несколько дней уже играете.

Информатика — это не только компьютеры, но и вычисления, которые происходят повсюду. Думайте как программист, и вы начнете подмечать вычисления и видеть возможности усовершенствовать действительность — масса шансов воплотить идеи в реальность.

Навыки для XXI века

Все, кто изучает информатику, получают бонус — осваивают новый фундаментальный тип мышления и способ решения задач. Этот тип мышления абсолютно необходим в новом мире, где высокие технологии повсеместны. Он называется **«вычислительное мышление»** и является большим преимуществом для тех, кто осваивает информатику, независимо от их будущей профессии. Эта идея получила огромный резонанс, и во многих странах вычислительное мышление добавили к чтению, письму и арифметике в качестве ключевого навыка, который нужно осваивать уже в начальной школе. Благодаря ему компьютеры теперь преобладают во многих сферах нашей жизни и меняют все, чем мы занимаемся, — от слушания музыки до торговли на бирже, от шопинга до науки. Вычислительное мышление дает нам возможность не только высказывать блестящие идеи, но и воплощать их в реальность.

В первый раз словосочетание «вычислительное мышление» использовал педагог и математик Сеймур Паперт. Он предложил обучать математиков совершенно новым способом — с использованием компьютеров. Однако благодаря информатике изменилась не только математика, но и вся наука. Ученый-информатик Дженнет Уинг заявила, что это самый важный компонент в изучении информатики, который надо использовать гораздо шире. Именно она популяризовала термин «вычислительное мышление». Компания Microsoft была настолько впечатлена ее аргументами и важностью поднятой темы, что предоставила Университету Карнеги–Меллон, где работала Уинг, грант

в несколько миллионов долларов на создание центра по изучению этого аспекта информатики и его влияния на другие науки.

Так что же такое вычислительное мышление? Это не «то, как думают компьютеры», хотя их все чаще программируют на его использование. Это набор разнообразных человеческих навыков для решения задач. Чтобы их приобрести, необходимо изучать природу вычислительных процессов. Кроме того, необходимы такие определенно важные навыки, как умение творить, ясно объяснять и работать в команде, — но их развивают практически все учебные предметы. Вычислительное мышление заимствует элементы из других типов мышления, например математического и научного. Однако в его основе лежат очень конкретные навыки решения проблем, такие как способность мыслить логически и алгоритмически, не упуская ни одной детали, а также умение находить эффективные способы что-нибудь сделать. Также важную роль играет способность понимать других людей. Информатика уникальна в том плане, что она объединяет все эти разнообразные навыки. Вместе они формируют мощный тип мышления, который меняет мир. Именно благодаря ему мы стали по-новому заниматься наукой, делать покупки, вести бизнес, слушать музыку, играть в игры — в общем, жить по-новому.

Алгоритмическое мышление

Алгоритмическое мышление лежит в основе вычислительного мышления. Оно позволяет находить решения задач нетрадиционным способом. Для специалиста по информатике решить задачу — это не просто получить ответ, например «42». И даже не добиться конкретного результата — например, «я решил судоку из сегодняшнего номера». Решения — это алгоритмы! *Алгоритм* — просто набор инструкций, которым необходимо следовать. Если точно их выполнять, вы получите собственно ответ на задачу («42») или добьетесь, чего хотите (например, решите судоку). Как только вы получите алгоритмическое

решение, вы будете решать подобные задачи вообще не задумываясь, просто «слепо» следуя инструкциям. При наличии такого алгоритма задачу, не углубляясь в суть программы, решит кто угодно. При этом не надо знать или понимать, как в конечном итоге работает алгоритм. Можно даже не иметь представления о том, что вы решаете судоку (и что такое вообще судоку). А это значит, что бездушная машина, компьютер, тоже будет механически следовать инструкциям и решать подобные задачи в любой форме. Именно так и работают компьютеры — они выполняют алгоритмы, написанные людьми.

Настоящая сила этой идеи в том, что следование алгоритму обеспечивает решения для целой группы задач, а не только для одного примера. Алгоритм для решения кроссвордов позволит решить много кроссвордов. Алгоритм для арифметических действий справится с любым расчетом. Когда мы воспринимаем задачи и решения таким образом, это называется **алгоритмическое мышление.**

Например, недостаточно знать, что 20 + 22 равно 42. Специалисту по информатике нужен алгоритм, который будет складывать два числа. На самом деле все мы в начальной школе изучаем этот алгоритм именно для того, чтобы решать примеры и не углубляться в его составление самим! Подобным образом, во все компьютеры встроена инструкция по сложению — вот насколько она важна! Компьютер действует только как калькулятор, только следуя инструкциям, которые указывают ему, как проводить вычисления. Компьютерная программа — это просто алгоритм или набор алгоритмов, написанных на языке, который понимает машина, — на *языке программирования*.

Изменить мир

Однако речь здесь идет не только о вычислениях. Алгоритмы можно использовать в самых разных целях. Мыслите алгоритмически, и у вас появится мощный способ изменить мир. Если

записать алгоритмы в виде программ, то они будут слепо выполнять все что угодно. Сегодня банки вместо людей используют алгоритмы, чтобы торговать ценными бумагами — покупать, продавать и получать миллионные прибыли. НАСА использует их, чтобы запускать корабли на Марс. Вы пользуетесь алгоритмами, чтобы слушать музыку и смотреть видео. Алгоритмы управляют самолетами, помогают хирургам и позволяют нам делать покупки, сидя у себя в гостиной или в вагоне поезда. Они водят машины и даже создают произведения искусства. Сейчас алгоритмы присутствуют во всех аспектах нашей жизни. Алгоритмы уже преобразили нашу жизнь и продолжают это делать. Поэтому важно понимать, что такое алгоритмическое мышление. Так же, как мы изучаем физику, чтобы понимать физический мир, и биологию — чтобы понимать живой мир, всем нам необходимо в какой-то мере освоить информатику, чтобы понять виртуальный мир, который тихо захватил нашу жизнь.

Научное мышление

Алгоритмическое мышление — это не просто способ решать задачи. Оно открывает новые пути для понимания мира. В традиционной науке используются эксперименты. Биологи ставят эксперименты на крысах и обезьянах, на клеточных культурах. Медики проводят испытания лекарственных препаратов. Физики ставят эксперименты над самим миром. Однако, если мыслить алгоритмически, возможен другой вариант. Если существует теория, объясняющая некое явление, будь то воздействие радиации на поверхность планеты, формирование экосистемы или развитие злокачественной опухоли, мы можем создать алгоритмы, работающие подобным образом. Мы можем создать **вычислительную модель** — программу, которая должна симулировать интересующие нас феномены, и проводить эксперименты на модели, а не в реальном мире. Если мы

правильно понимаем явление, то программа будет вести себя как объект моделирования. Если этого не произойдет, значит, с нашей теорией что-то не так. Обдумывая, что пошло не так, мы обнаружим, что́ надо изменить в теории, и, таким образом, будем лучше понимать явление. Вполне вероятно, что изучение модели выведет нас на новые предположения, которые можно проверить уже в реальном мире.

Вычислительное мышление

Вычислительное мышление подразумевает не только поиск решений в виде алгоритмов. Это целый набор приемов, который обеспечивает нам эффективный способ улучшения жизненных условий и осмысления мира. Но мы не будем погружаться в специфические термины, а продемонстрируем эти методы на примере задач как серьезных (например, помощь инвалидам), так и развлекательных (игры, головоломки и фокусы).

Глава 2

В поисках способа говорить

Одним из самых тяжелых патологических состояний, какие только можно вообразить, является синдром «запертого человека». Человек в таком состоянии полностью парализован и в лучшем случае в состоянии только моргать. Разум заключен в тюрьму бесполезного тела. Человек воспринимает все вокруг, но не может передавать информацию. Тем, кто хотел бы помочь людям с таким синдромом, очевидно, нужно учиться на медиков. Но может ли что-нибудь сделать программист?

Сидром «запертого человека»

Синдром «запертого человека» — это полный паралич тела после инсульта. Вы продолжаете думать, видеть, слышать. Вы так же разумны, как и прежде. Это может случиться с каждым. Лечения этого заболевания нет, поэтому максимум, что могут сделать медики, — позаботиться об удобстве пациента. Но возникает важный вопрос: как помочь пациентам с синдромом «запертого человека» «разговаривать». Как им общаться с врачами, семьей и друзьями? Очевидно, специалист по информатике мог бы изобрести новую технологию, которая была бы полезна в этой ситуации. Однако благодаря вычислительному мышлению мы можем предложить способ гораздо лучше, чем просто «полезная технология».

«Скафандр и бабочка» — невероятно жизнеутверждающая книга. Это автобиография Жан-Доминика Боби, которую он написал после того, как очнулся в больнице полностью парализованным. Он рассказывает о жизни с синдромом «запертого человека». То есть у него был способ общения, который позволил не только разговаривать с медиками, друзьями и семьей, но и написать книгу. Боби сделал это, вообще не прибегая к технике. Но как?

Представьте себя в его положении — очнулись на больничной койке. Как вы могли бы общаться? Как могли бы написать книгу? Только человек с ручкой и бумагой смотрит на вас, готовый записывать слова. Вы из тех, кому повезло, — вы можете моргать одним глазом, но это все. Это единственное движение, которое вам доступно. Значит, разговаривать вы не в состоянии. Однако вы видите и слышите.

Теперь представьте, что вы врач такого пациента и вам необходимо придумать способ общения с ним.

Просто как A, B, C

Вам нужно условиться о способе превратить моргание (все, что доступно пациенту) в буквы. Возможно, сначала вам придет в голову такой вариант: когда он моргнет раз, это будет означать «A», два раза — «B» и так далее. Тогда помощнице останется посчитать, сколько раз моргнул пациент, и записать соответствующие буквы.

Предложив такую идею, мы уже рассуждаем как программисты. То, чем мы занимаемся, лежит в основе вычислительного мышления — это **алгоритмическое мышление.** Мы придумали серию шагов, которым могут следовать больной и его помощница, чтобы гарантированно передать и понять нужные буквы. В информатике такой способ коммуникации называют *алгоритмом.* Он представляет собой серию шагов, которые необходимо пройти в заданном порядке, чтобы достичь определенной цели

(в данном случае — передать буквы и слова). **Алгоритмическое мышление** необходимо, чтобы разрабатывать алгоритмы для решения задач.

Красота алгоритмов в том, что им следуют, не имея представления, что́ именно они значат. В случае с нашим алгоритмом помощница предположительно знает, что и для чего она делает, но книга все равно была бы написана, даже если бы она ничего не понимала. Все, что нужно делать, — считать моргания и записывать буквы в соответствии с полученными инструкциями. Мы могли бы дать помощнице таблицу, чтобы сверять по ней буквы, и тогда работа выполнялась бы без какого-либо ее осмысления вообще. Красота алгоритмов заключается в возможности действовать механически, и в этом их смысл — ведь компьютеры тоже слепо выполняют инструкции. Это умеют абсолютно все компьютеры.

Наш алгоритм общения на деле состоит из двух частей. Одну часть выполняет Боби (моргнуть нужное количество раз), а другую — помощница (сосчитать, сколько раз моргнул Боби, и записать соответствующую букву, когда моргание прекратится). Более того, в информатике есть специальное название для алгоритма, при помощи которого делятся между собой информацией два человека или компьютера, — он называется *протокол*. Если оба человека выполнят свою часть протокола, то слова, которые задумал Боби, окажутся записанными на бумаге. Если кто-то сделает ошибку — например, собьется со счета и таким образом отойдет от протокола, — то сообщение не будет доставлено. В компьютерах хорошо то, что они не делают таких ошибок, каждый раз точно выполняют инструкции. Коль скоро инструкции верны, машины-то уж точно их верно выполнят.

Алгоритмическое мышление — это особый род решения проблем, при котором вы не просто находите один ответ (например, что именно хотел сказать Боби, когда очнулся после инсульта). Вы находите решение в виде шагов, которые могут выполнить другие (в том числе компьютер), — и тоже получить

ответ. Мы только что нашли подобное решение для Боби, благодаря которому понимаем не только то, что он пытается сказать в данный момент. Этот способ позволяет нам (и кому угодно) в любой момент выяснить, что он хочет сказать. Но судя по всему, процесс пойдет довольно медленно. Может быть, есть способ получше. Придумывать более удачные, эффективные решения — это тоже часть **алгоритмического мышления.**

Как это сделал Боби?

У Боби был улучшенный способ, а точнее — алгоритм, который он описывает в своей книге. Вспомним, что у помощницы нет проблем с речью, и это можно использовать. Алгоритм работал так: помощница читала вслух алфавит («А… В… С…»), и, когда звучала нужная буква, Боби моргал. Тогда помощница записывала ее — и опять начинала сначала. Попробуйте это вместе с другом — передайте таким образом свои инициалы. А теперь представьте, что это единственный способ общения с людьми. Остается надеяться, что вас зовут не Яна Яковлевна Яблочкина и не Ярослав Яромирович Якубович!

А теперь представьте, что так проходит вся жизнь. Что так вы вынуждены разговаривать с семьей и друзьями. И если вы хотите, чтобы открыли шторы или переключили телеканал, то придется просить об этом таким способом.

Попробовав, вы, вероятно, осознаете, что для эффективного применения этого метода нужно решить еще кое-какие проблемы. А после нескольких попыток вам, весьма вероятно, придет в голову способ улучшить алгоритм. Что вы можете предложить?

Проверяем детали

Нетрудно осознать, что придется иметь дело не только с буквами алфавита. Нам также понадобятся пробелы, цифры, точки и так далее. Их необходимо добавить к списку букв, который

использует помощница. Вероятно, есть способ и получше, чем зачитывать длинный список. Например, сначала задать вопрос: «Это буква?» Если ответ положительный, то будем продолжать как раньше. Если нет, переходим к другим символам. Звучит знакомо? Это та же идея, благодаря которой в компьютерах используются разные наборы символов.

Еще одна проблема, требующая решения: что делать, если человек моргнет по ошибке? У нас должен быть способ сказать: «Проигнорируйте последний раз и начинайте читать буквы с начала». Но так, чтобы не пришлось передавать эту фразу по буквам! Подобным образом, если вы сделали ошибку, нужно найти способ вернуться назад. Нам нужен код, который означает «отменить». Возможность отменить действие — важная часть любого алгоритма с участием людей, так как люди делают ошибки. Например, условимся, что для этого надо быстро моргнуть два раза. Или придумайте что-нибудь получше. Вполне вероятно, что вы обнаружите другие проблемы, требующие решения?

В теории и на практике такая проверка или **оценка** работы алгоритма является важной составляющей вычислительного мышления**.** Если мы придумали новый алгоритм, его работу надо очень тщательно проверить. Программисты на оценку программ (то есть алгоритмов для компьютеров) тратят больше времени, чем на их создание. Очень легко ошибиться в какой-то мелочи или забыть о возможной ситуации, с которой должен справиться алгоритм. Но смысл алгоритма в том, что он работает всегда, что бы ни случилось.

Алгоритмическое мышление подразумевает, что мы обдумываем детали и находим решения для возникающих проблем. Мы осознаем, что есть много способов сделать одно и то же, а потом предлагаем улучшенные варианты для конкретной ситуации. Также заметим, что одна из упомянутых выше задач связана с характерной для человека особенностью — свойством ошибаться. Теоретически наше решение работает, надо только моргнуть в нужный момент! И мы могли бы высокомерно

заявить, что надо совершать определенные действия, а не получилось — сами виноваты. На практике не всегда моргаешь, когда нужно. И лучше все-таки решить задачу так, чтобы алгоритм работал для людей. В конце концов, мы пытаемся помочь человеку, а не машине! Вычислительное мышление связано еще и с пониманием того, что такое человек.

Улучшаем метод

Что дальше?

Мы могли бы немного ускорить процесс общения для пациента с синдромом «запертого человека», осознав, что порой уже на половине слова можно догадаться, что имеется в виду. Например, если у вас получилось «а-н-т-и-л», с большой долей вероятности можно утверждать, что нужное слово — «антилопа». Значит, поменяем правила так, чтобы помощница высказывала подобные догадки. Кроме того, надо найти способ сказать «нет», если догадка не верна. Например, такое правило: моргнуть, если слово угадано, и не моргать — если нет. Именно по этому принципу работает функция *предиктивного ввода текста* в телефоне, то есть используется алгоритм для решения очень похожей задачи. То же самое делают поисковые движки, когда вы набираете свой запрос.

Помощники Боби действительно использовали вариант предсказания текста, что и описано в его книге. Он также отмечает, что его очень раздражало, если люди пытались угадать его мысли, не условившись с ним о способе подтверждения. Отсутствие навыков вычислительного мышления у собеседников Боби приводило к тому, что он очень расстраивался, пытаясь «сказать» им, что они ошиблись, а собеседники были уверены, что догадались правильно. Представим, например, что мы продолжаем разговор о животных и я передал буквы «б-а-р-с». Какова будет ваша догадка? Что слово уже закончилось и это слово — «барс»? Нет. Я хотел сказать «барсук».

Возможно, вам тоже пришла идея об угадывании целого слова, ведь вы пользовались предиктивным вводом текста в телефоне. Если так, это значит, что вы только что использовали еще один навык вычислительного мышления — **сопоставление с образцом.** Часто задачи, в сущности, повторяют то, что вы уже видели в другой ситуации. Если у вас уже есть решение для определенной проблемы, то есть смысл использовать его повторно. **Сопоставление с образцом** — навык, который позволяет понять, что новая ситуация по сути повторяет уже известную вам, и увидеть, что можно использовать старое решение.

Алгоритмы обеспечивают такого рода общее решение. Мы можем повторно использовать технологию предиктивного ввода текста, потому что у телефона и помощницы Боби одна и та же проблема. Телефон должен догадаться, какие слова набирает по буквам пользователь, а помощница — какое слово передает по буквам пациент с синдромом «запертого человека». Как только мы осознали это сходство, любое решение, найденное для первого случая, реально использовать для второго. Еще лучше, если мы увидим, что обладаем решением, которое подходит для множества разных задач, сделаем описание алгоритма с самого начала и будем использовать его при необходимости. Это называется **обобщением** алгоритма. **Обобщение** — очень мощный метод вычислительного мышления.

В самом широком смысле можно считать, что в случае Боби мы занимаемся передачей информации. В любой ситуации, когда есть необходимость передать информацию, используется общий алгоритм. Программисты создают коллекции алгоритмов для разного рода задач, чтобы при необходимости выбрать наиболее подходящий. Например, *азбука Морзе* тоже алгоритм передачи информации. Используя разную последовательность точек и тире (в нашем случае — долгое или быстрое моргание), обозначают разные буквы. Этот алгоритм изобрели, чтобы передавать сообщения по телеграфу, но, вероятно, получится использовать его и здесь. Мы еще вернемся к этой идее.

В еще более широком смысле мы вправе представить нашу задачу как поиск очередной доли информации (следующая буква). И видимо, мы сумеем обобщить наш алгоритм настолько, что он позволит искать что угодно. Ниже мы вернемся и к этой идее.

В порядке популярности

Боби предложил другой способ улучшить алгоритм ABC. До того, как оказаться на больничной койке, он был главным редактором французского женского журнала *Elle* и имел хорошее представление о языке. Например, ему было известно, что E — самая распространенная буква (в английском и французском). Поэтому Боби попросил, чтобы буквы зачитывали в порядке их популярности — то есть *частотности*. В английском этот порядок таков: E, T, A, O… Во французском, на котором говорил Боби, это E, S, A, R… Боби, соответственно, использовал французский порядок. Таким образом, помощница быстрее доходила до распространенных букв.

Похожий трюк использовался веками, чтобы расшифровать секретные коды. Он называется *частотный анализ*. Алгоритм для использования частотности букв был изобретен арабскими учеными около 1000 лет назад. Марию Стюарт обезглавили, потому что сэр Фрэнсис Уолсингем, начальник разведки королевы Елизаветы I, лучше нее владел вычислительным мышлением. Но это уже другая история. Идея Боби использовать частотный анализ — это пример и **сопоставления с образцом**, и **обобщения**. Задачи трансформируются, и решения для них используются повторно. Осознав, что расшифровывание кодов и угадывание букв — процессы схожие, мы видим, что частотный анализ, изобретенный для одного, пригоден для другого.

Насколько это быстро?

Давайте вернемся к алгоритму Боби, который мы определенно усовершенствовали. Новый способ должен быть лучше

изначальной идеи — моргать разное количество раз для разных букв. Однако напрашивается вопрос: как быстро это будет — сколько времени уйдет, чтобы написать книгу? Удалось ли найти наилучший способ или можно предложить более быстрый алгоритм, который облегчит написание книги?

Нам необходимо определить эффективность алгоритма. Проведем эксперимент и применим **научное мышление.** Например, следующим образом: несколько раз определим время, которое уходит на передачу какого-то отрывка с каждым алгоритмом и с разными участниками, и выясним, в каком случае все было в среднем быстрее. Однако на это уйдет очень много времени и сил. Есть способ и лучше.

Можно прибегнуть к **аналитическому мышлению.** В этом случае необходимо сделать простые вычисления. Например, давайте учитывать не время, а сделанную работу. Если подсчитать, сколько букв алфавита произносит помощница, то мы всегда определим потраченное время. Просто надо знать, сколько времени уходит на произнесение одной буквы, и умножить это время на количество букв. Мы только что произвели действие, которое называется **абстрагированием.** Это еще один элемент вычислительного мышления, который применяется, чтобы упростить задачи и облегчить написание программ. **Абстрагирование** — просто длинное слово, которое подразумевает, что некоторые подробности скрывают или игнорируют. Мы проигнорировали такую деталь, как точное время, потраченное на всю книгу, и вместо этого подсчитали произнесенные буквы. «Число произнесенных букв» — это **абстракция** реально потраченного времени. Такой принцип очень часто используется в вычислительных процессах, чтобы упростить их работу.

Как же нам выяснить, сколько букв надо произнести? Для этого нужно задать несколько вопросов. Самый простой звучит так: сколько это будет *в лучшем случае?* Каково минимальное количество букв, которое должна произнести помощница, чтобы получилась книга? Рассмотрим и *худший случай.* Если не повезет,

то насколько? Наконец, рассмотрим *средний вариант* и таким образом получим реалистичную оценку необходимой работы. Давайте чисто теоретически представим, что нам нужны только буквы алфавита, без цифр и знаков пунктуации. И проанализируем наш простой алгоритм, в соответствии с которым помощница говорит: «A, B, C…»

В лучшем случае вся книга будет состоять только из «A»: «AAAA…» (возможно, выражая боль автора). Чтобы общаться при помощи одной буквы «A», достаточно сказать «A» один раз (ответить на один вопрос), и ответ будет получен. Здесь мы снова используем **абстракцию** — сначала анализируем, что будет, если посчитать только одну букву, и игнорируем всю книгу — по крайней мере для начала. Умножьте наш ответ для одной буквы на количество букв в книге и получите упомянутый лучший случай.

В худшем случае (для латинского алфавита), при котором кто-нибудь, например, все время жужжит («ZZZZ…»), потребуется 26 вопросов для каждой буквы. Итак, мы определили границы, в которых будет происходить передача любой информации. У нас никогда не получится лучше, чем при варианте с одной буквой, и хуже, чем со всеми 26.

Оценка будет точнее, если учесть среднее количество вопросов на каждую букву, то есть средний случай. Сделать это не так трудно. В длинном сообщении на каждую «A» где-нибудь еще придется «Z», на каждую «B» найдется «Y» и так далее. Это значит, что в среднем во всей книге на каждую продиктованную букву надо будет задать 13 вопросов. Умножьте число букв в книге на 13, и вы получите примерную оценку работы при ее написании. Умножьте это на среднее время, которое уходит у помощницы, чтобы произнести букву, и вы получите время, необходимое, чтобы написать книгу.

Отметим, что мы снова оцениваем наш алгоритм — но на сей раз нас интересует не то, действует ли он вообще, а то, насколько быстро он действует. У алгоритма оценивают много разных

аспектов, но надежность и эффективность — два важнейших для **оценки**.

Изменение, внесенное Боби, — сначала спрашивать о распространенных буквах — улучшает ситуацию. Вероятно, получится уложиться в 10–11 произносимых букв. А с учетом частотности, мы рассчитаем это точнее. Частотность можно уточнить или определить самостоятельно. Возьмите отрывок из любимой книги и посчитайте, сколько раз появляется каждая буква. Потом расположите буквы по порядку, начиная с самой распространенной, и посчитайте вероятность их появления. Средний случай — это число букв, которое необходимо произнести для угадывания одной буквы, вероятность появления которой равна 50%.

Итак, частотный анализ привел к улучшениям, но не слишком значительным, и в худшем случае для выяснения одной буквы все равно надо задать 26 вопросов. Но каждый специалист по информатике знает, что можно существенно улучшить этот процесс. Любая буква выясняется всего за пять вопросов! Гарантированно! И это не средний случай, а худший! Знаете, какие пять вопросов надо задать?

20 вопросов?

Уложитесь в пять

Смогли вы ответить на этот вопрос или нет, я гарантирую, что вы знаете, о каких вопросах идет речь. Чтобы вспомнить о них, нужно рассмотреть другую задачу.

Давайте сыграем в игру «20 вопросов». Это детская игра, в которой водящий задумывает известного человека, а вы пытаетесь догадаться, кто это, задавая вопросы. Изюминка в том, что отвечать следует только «да» или «нет». Сыграйте в эту игру с другом и обратите внимание, какие вопросы вы задаете. Представим, как может пойти игра.

«Вы женщина?» — «Нет».
«Вы живы?» — «Нет».

«Вы были кинозвездой?» — «Нет».
«Вы жили в Британии?» — «Да».
«Вы были писателем?» — «Да».
«Вы жили в XX веке?» — «Нет».
«Вы жили в XIX веке?» — «Нет».
«Вы Шекспир?» — «Да».

Вероятно, играя, вы задавали похожие вопросы. Очень маловероятно, что вы сразу начали спрашивать: «Вы Аристотель? Вы Джеймс Бонд? Вы Мария Кюри?» Так вы никогда бы не нашли ответ за 20 вопросов. До подобных формулировок дело обычно доходит в конце, когда вы практически уверены, что знаете, кто это (как мы только что показали). Скорее, вы начали с вопроса вроде «Вы женщина?».

Почему это хороший вопрос для начала? Да потому, что он отметает половину возможных вариантов при любом ответе. Если вы спросите «Вы королева Англии?», то в случае успеха отбросите миллионы других вариантов, а в случае (более вероятного) неуспеха — только одного человека. Чтобы сразу угадать, вам должно повезти не меньше, чем выигравшему в лотерею. Значит, секрет игры «20 вопросов» — задавать вопросы так, чтобы каждый раз отбрасывать половину людей, каким бы ни был ответ.

Насколько это эффективно?

Задавать вопросы, которые оставляют половину возможных ответов, — лучше, чем называть конкретное имя, но насколько? Давайте предположим, что я изначально задумал кого-то одного из миллиона. Если после каждого вопроса отметать половину людей, сколько вопросов понадобится? После первого останется 500 000 человек, после второго — 250 000... После десяти вопросов от исходного миллиона останется примерно 1000 человек (рис. 1). Продолжаем... После следующего вопроса осталось 500, потом 250, 125... и на двадцатом вопросе остается

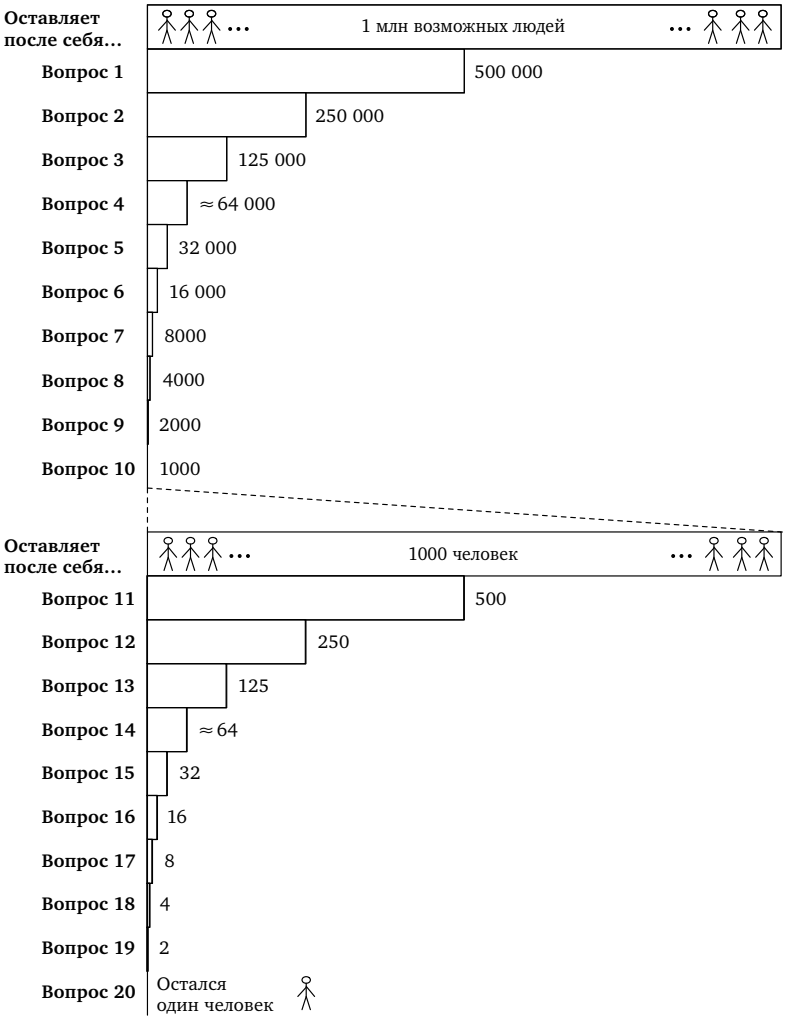

Рис. 1. Если после вопроса остается половина вариантов, то реально сократить число возможных ответов с миллиона до одного всего за 20 вопросов

один возможный человек. Если вам удастся каждый раз точно задавать вопрос, после которого останется половина ответов, то вы гарантированно выиграете. И всегда это будет 20 вопросов.

Все это, конечно, **алгоритмическое мышление.** Мы пытались разработать алгоритм для игры «20 вопросов». Однако до конца решить задачу не удалось — было непонятно, как определить нужные вопросы. Это остается вашей задачей во время игры. Здесь используется еще один трюк вычислительного мышления — **декомпозиция** (разложение на части)**.** Надо разделить проблему на части, чтобы сосредоточиться на каждой отдельно. Пока у нас получилось найти общую стратегию. Подобрать конкретные вопросы, которые оставят половину вариантов, — отдельная проблема.

Декомпозиция — популярная стратегия для решения задач и жизненно важный инструмент информатики. Задачи, которые надо решать при составлении программ или разработке процессов (например, для вашего ноутбука или телефона), имеют гигантские масштабы. Современные компьютерные микросхемы сложнее, чем дорожная сеть всей планеты Земля. Представьте, что вы пытаетесь решить такую задачу в один прием. Это можно сделать, только разложив ее на части и работая над ними отдельно.

Декомпозиция полагается на **абстракцию** — сокрытие деталей. Здесь мы абстрагируемся от конкретных вопросов и думаем только о том, какого типа вопросы надо задавать. Мы также использовали **декомпозицию**, когда размышляли, насколько эффективным был наш изначальный алгоритм. Чтобы выяснить, каким образом можно написать книгу, мы разложили одну задачу на две: выяснить, как передавать отдельные буквы, и провести всю необходимую работу с помощью полученного решения.

Новый алгоритм

Итак, если правильно задавать вопросы, то в худшем случае понадобится только 20 вопросов, чтобы угадать задуманного человека из миллиона возможных вариантов. Вспомним теперь,

что хватит 13 вопросов (в худшем случае — 26), чтобы определить одну из 26 букв алфавита. «Да/нет» не отличается от «моргнуть / не моргать». А спрашивать «Это A? Это B?» — примерно то же самое, что спрашивать «Вы Микки-Маус?» или «Вы Нельсон Мандела?». Вы точно так же пытаетесь выяснить, о какой из многих вещей я думаю. Это опять-таки та же самая задача, что и предиктивная система набора текста в телефонных сообщениях!

Но если это та же самая задача, то и соответствующая ей стратегия должна обеспечить нам более удачное решение, чем уже найденное. Здесь мы снова используем **сопоставление с образцом** и **обобщение.** Мы преображаем задачу, чтобы повторно использовать решения. Каков эквивалент решения, которое оставит только половину вариантов, применительно к алфавиту? Сначала, наверное, имеет смысл спросить: «Это гласная?» — но как будут выглядеть остальные четыре вопроса? Каждый раз оставлять только половину вариантов из алфавита? Напрашивается такой первый вопрос: «Это между A и M?» Если ответ утвердительный, то потом мы спрашиваем: «Это между A и F?» Если ответ отрицательный, мы спрашиваем: «Это между N и S?» — и так далее. Таким образом мы гарантированно доберемся до любой буквы алфавита, которую задумал человек, всего за пять вопросов, как это показывает *дерево решений* на рис. 2. Начните сверху диаграммы и двигайтесь вниз в соответствии с ответами «да/нет».

В этот момент нужно подключить еще один компонент **алгоритмического мышления.** Необходимо прояснить все детали, потому что здесь можно запутаться. Спрашивая «Это между A и M?», надо уточнить, входит ли «M» в этот промежуток (входит).

Попробуем еще больше усовершенствовать эту технику, используя частотный анализ. Поскольку букв только 26, реально добраться до «E» и других распространенных букв быстрее чем за пять вопросов. Попробуйте сделать дерево решений, которое это обеспечит. Кроме того, можно использовать принцип

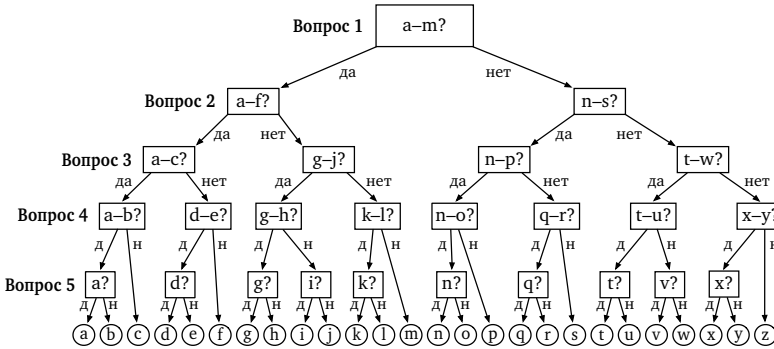

Рис. 2. Дерево решений с вопросами, необходимыми, чтобы добраться до любой буквы алфавита за пять и менее вопросов

предиктивного набора текста, чтобы предугадывать набранные не до конца слова. Все подобные решения из более ранних алгоритмов применимы и здесь. Мы повторно используем готовые решения.

Коды для букв

Дерево решений представляет собой совсем другой подход. Если мы примем «да» и «нет» или «моргнуть» и «не моргать» за 1 и 0, тогда дерево решений определит двоичную последовательность, которую должен усвоить больной с синдромом «запертого человека», чтобы обозначить каждую букву (рис. 3).

Таким образом, чтобы ускорить процесс, можно отказаться от вопросов. Человек, передающий информацию, проходит определенную последовательность для каждой буквы, а другой человек — записывает. Таким образом, обозначая морганием код 0110 (не моргать, моргнуть, моргнуть, не моргать), выражаем букву «P». Соответственно, дерево решений преображаем в *таблицу поиска*, как на рис. 4. Тому, кто хочет общаться с больным, можно дать либо дерево решений, либо такую таблицу. В сущности, мы только что изобрели код для общения, похожий на азбуку Морзе. И снова очевидно, что наша задача, в сущности,

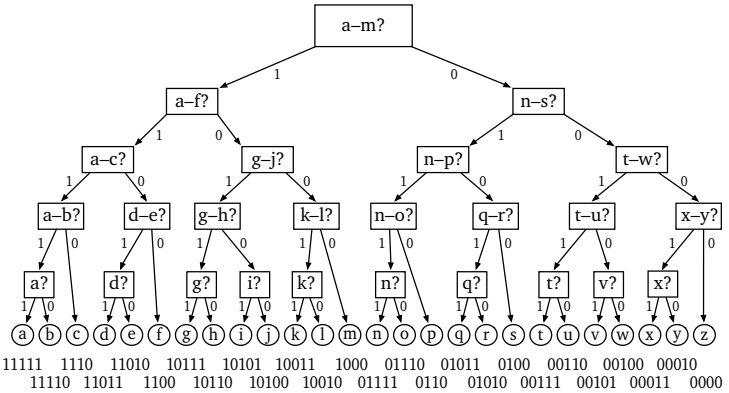

Рис. 3. Коды для каждой буквы в дереве решений

аналогична той, которую пытался решить Сэмюэл Морзе, чтобы передавать информацию с помощью телеграфа. Точки и тире соответствуют нашем единицам и нулям или вариантам «моргнуть» и «не моргнуть». И снова мы применяем **обобщение.**

Однако не стоит выбирать решение второпях. Детали играют большую роль. Если не задавать вопросов, как узнать, что человек передает информацию? Если он не моргает, что это

Код	Буква
11111	a
11110	b
1110	c
11011	d
11010	e
1100	f
10111	g
10110	h
10101	i
10100	j
10011	k
10010	l
1000	m

Код	Буква
01111	n
01110	o
0110	p
01011	q
01010	r
0100	s
00111	t
00110	u
00101	v
00100	w
00011	x
00010	y
00000	z

Рис. 4. Таблица поиска с двоичным кодом для каждой буквы

значит — он уснул или просто ничего не говорит? Как узнать, что он начал передавать буквы? Сколько времени продолжается «неморгание»? Небольшое изменение привело к необходимости решить много новых проблем. Сэмюэл Морзе их решил. В азбуке Морзе с этой целью используются три символа, а не два: точки, тире и тишина. Продолжительность каждого элемента точно определена. Какой бы долгой ни была точка, пауза между точками и тире одинаковая. Между буквами она в три раза дольше точки, между словами — в семь раз. Это обеспечивает структуру, которую мы потеряли, отказавшись от вопросов.

Решение в виде кода отлично подошло для телеграфа, и его вариант лежит в основе взаимодействия компьютеров в сети. Но можно ли сказать, что это решение больше подходит человеку с синдромом «запертого человека», — вопрос спорный. Машинам легко иметь дело с точными промежутками времени, а людям это гораздо сложнее, чем просто задать вопрос.

Выбираем лучшее решение

Алгоритмы поиска

Наше решение для игры «20 вопросов» можно использовать, чтобы помочь пациенту с синдромом «запертого человека» разговаривать, потому что задача, в сущности, не меняется. Это *задача поиска*: у нас есть серия предметов и надо найти один конкретный. Решения для этой задачи называются *алгоритмами поиска*, и они обеспечивают беспроигрышный способ что-нибудь найти. Первый подход, проверка всех вариантов по очереди («Это А?», «Это В?..», «Это певица Адель?», «Это Джеймс Бонд?..»), — алгоритм под названием *линейный поиск*. Порой это лучшее, что есть в вашем распоряжении. Например, если вы были свидетелем ограбления и участвуете в опознании преступника из нескольких людей, линейный поиск будет для вас оптимальным: рассмотрите по очереди каждое лицо, пока не увидите в ряду человека, который совершил преступление.

Линейный поиск хорошо работает и тогда, когда вещи, среди которых вы ведете поиск, никак не упорядочены. Если вы ищете носок, который может быть в любом ящике комода, начните сверху и последовательно проверяйте ящики один за другим.

Другой алгоритм включает вопросы, после ответа на которые остается половина вариантов: «Эта буква стоит раньше N?», «Это женщина?». Найти подобные вопросы — общая стратегия решения задач под названием *«разделяй и властвуй»*. Если вы найдете такое решение, ответ, вероятно, будет получен очень быстро. Почему? Потому что, как мы видели, если несколько раз сократить число вероятных ответов вполовину, можно очень быстро прийти к одному варианту — гораздо быстрее, чем если бы мы проверяли последовательно пункт за пунктом. Заметим, что здесь мы снова используем **обобщение.** Самый простой алгоритм поиска по принципу «разделяй и властвуй» называется *бинарным поиском*. Представьте, что все предметы, среди которых вы ищете нужный, стоят по порядку и самый маленький — на одном конце, а самый большой — на другом. В ходе бинарного поиска вы подходите к предмету, который расположен посередине, и проверяете, лежит ли нужная вам вещь до или после него. Затем вы отбрасываете ненужную половину и повторяете ту же операцию с оставшейся. Вы делаете это до тех пор, пока не остается один предмет — тот, который вы ищете. Возможно, примерно так вы поступаете, когда нужно найти фамилию в толстом телефонном справочнике. Конечно, вы не будете начинать с первой страницы и проверять каждое имя по очереди, пока не найдете нужное!

Кроме этих двух, есть еще много алгоритмов поиска. Например, каким образом поисковая система вроде Google просматривает каждую веб-страницу на планете за доли секунды? Она использует еще более продвинутый алгоритм!

Чтобы применить алгоритм поиска, необходимо задействовать **абстрагирование.** Мы абстрагируемся от деталей конкретной задачи и смотрим, нельзя ли свести ее к задаче поиска,

и тогда наш алгоритм поиска становится готовым решением для многих задач. Можно подойти к этому с другой стороны: как только мы придумаем стратегию выигрыша за 20 вопросов, то сумеем обобщить это решение до алгоритма «разделяй и властвуй» — у нас есть общая стратегия, которая работает и для других задач. **Абстрагирование** и **обобщение** часто неразрывно связаны.

Делаем жизнь Боби лучше

Значит, надо было устроить так, чтобы помощница задавала Боби вопросы, после которых исключалась бы половина из возможных вариантов. Представьте себе: придется задать в худшем случае пять вопросов вместо 13 в среднем, помноженные на все количество букв в книге. И речь идет не только о книге, но и о разговорах с друзьями и родственниками, докторами и медсестрами. Если бы он был немного знаком с информатикой, насколько легче могла бы стать его жизнь!

Главное — алгоритмическое мышление

Здесь стоит отметить, что пока мы вообще не учитывали технологии. Речь шла исключительно о двух «беседующих» людях. Теперь, когда у нас есть хороший способ, то есть хороший алгоритм, озаботимся тем, как его автоматизировать при помощи подходящих технических средств. Мы могли бы использовать *систему управления с помощью движения глаз,* которая распознает моргание, или *шапочку с электродами,* которая улавливает, «да» у человека в мыслях или «нет». Но дело в том, что, какую бы технику мы ни использовали, ей все равно потребуется алгоритм поиска. Если выбрать его неверно, то при всех ее преимуществах общение все равно пойдет медленно — придется задавать 13 вопросов вместо пяти. И нет никакой разницы, будет ли помощником компьютер или человек. Если сначала

не продумать алгоритм, система может оказаться мучительно медленной. Вычислительная техника — это не только технологии. Это и вычислительное мышление, которое необходимо, чтобы найти хорошее решение.

Еще важнее — понять человека

Итак, нет сомнений, что жизнь Боби могла бы стать лучше, если бы вычислительное мышление применялось активнее. Но не будем торопиться с выводами. Возможно, мы неправильно поняли ситуацию. Есть вероятность, что в этом случае он никогда бы не написал книгу и его жизнь превратилась бы в еще больший ад. Почему? Мы начали не с технологий, а с информатики. Возможно, надо было начинать с человека. Удалось ли нам учесть главное?

В качестве показателя успеха, или нашей **абстракции,** мы использовали количество заданных вопросов. Задавать вопросы — задача помощницы, и это нетрудная работа, хотя и нудная. А что, если Боби было трудно моргать? При его решении надо было моргать один раз на каждую букву. Наш алгоритм типа «разделяй и властвуй» требует, чтобы он моргал пять раз. Умножьте это на всю книгу. Не исключено, что наше решение сделало бы его задачу в пять раз сложнее! Но возможно, моргать было легко и наш алгоритм действительно лучше. Мы не знаем ответа, потому что не задали вопрос. А стоило бы сначала спросить. Боби не рассказал об этом в книге, и у каждого может быть свой ответ на этот вопрос. Поэтому и надо начинать с человека.

Более того, решение Боби понятно любому. Наше же относится к более сложным и, вероятно, потребует объяснений. И объяснять этот метод будет не Боби. Думать о людях — это важно.

Описанная ситуация выдвигает на первый план еще один необходимый аспект **оценки** алгоритма. Мы должны ответить на вопрос: легко ли использовать наш алгоритм, не делая

ошибок, и останутся ли у людей хорошие впечатления? Это необходимо сделать, даже если алгоритм выполняет компьютер, а люди взаимодействуют с программой. Так мы учитываем *удобство использования* и *восприятие пользователем* алгоритма. Подобная **оценка** в конечном итоге должна включать тестирование решений с участием реальных людей. И чем быстрее это делается, тем лучше.

Ему подошло

Про решение Боби одно известно точно: оно подошло для его целей. В конце концов, так он написал целую книгу. Возможно, помощница не только записывала его слова, но и открывала шторы, разговаривала с ним об окружающем мире или просто давала немного человеческого тепла. Возможно, вся книга нужна была для того, чтобы у Боби был постоянный собеседник, который получал за общение с ним деньги от издательства!

Таким образом, важно не только то, что коммуникационный алгоритм послужил созданию книги. Сама книга помогла удовлетворить глубинную потребность в непосредственном общении с другим человеком. Если бы человека заменили техникой, то Боби лишился бы того единственного, что поддерживало в нем жизнь.

В то же время, возможно, если бы он мог говорить с компьютером, то попал бы с больничной кровати в виртуальный мир и стал бы посылать сообщения друзьям, пользоваться социальными сетями, управлять аватаром, а однажды — даже версией себя в виде робота, который передвигается в реальном мире… Возможно, все это улучшило бы ситуацию.

Значит, прежде всего необходимо определить, чего на самом деле хочет человек и в чем он нуждается в первую очередь. В ситуации, когда удобство метода обретает крайне большое значение, надо все устроить так, чтобы пользователь с самого начала активно участвовал в процессе. Мы называем это