Как решается сложная задача.

Существует один общий подход к поиску решения сложной задачи, независимо от того, из какой она области: математическая, физическая или программисткая. Выражается этот подход в трех простых предложениях:

1. Определим тип задачи.
2. Вспомним, какими методами нам или кому-нибудь другому доводилось решать задачи такого типа.
3. Попробуем применить эти методы к нашей задаче.

Этот подход кажется логичным и разумным. Ведь большинство задач, с которыми мы сталкиваемся, кем-то уже решены, кто-то уже знает, как ответить на заинтересовавший вопрос, ответ уже есть в совокупной базе знаний человечества. Но накопленное общечеловеческое знание – это достаточно сложная штука. Оно не так доступно, как хотелось бы в силу своей огромности. Существуют и другие причины, по которым конкретный человек, сталкиваясь с конкретной задачей, не может найти готового решения. Поэтому несмотря на то, что человечество, как целое, знает и умеет уже довольно много, отдельный человек часто встречается с ситуацией неопределенной и следовательно творческой. А в такой ситуации наш алгоритм из трех предложений, уже не работает.

Кроме того если взять действительно интересную задачу, то окажется, что определить, к какому типу она относится довольно сложно. И часто задача будет относиться не к одному типу, а к нескольким. Например, это может быть задача комбинаторного характера с применением графов, или это может быть задача моделирования физических процессов с использованием графики и методов численной математики.

Если бы была возможна исчерпывающая классификация задач, с конечным количеством классов, четко отделимых друг от друга, то это бы означало ограниченность программисткой науки. Программирование, как наука закончилось бы в момент написания последнего алгоритма на последний неразрешенный класс задач.

Вроде бы нет ничего страшного, просто в таких задачах мы имеем дело со сложными типами, и все равно можно действовать по той же схеме, то есть последовательно выполнять действия 1, 2, 3.

К сожалению не получается. Во-первых, сложных типов можно сконструировать огромное количество, а чем типизация обширнее, тем сложнее выполнять пункт первый. Во-вторых, чем больше типов, тем сложнее в них ориентироваться, а в третьих, тем сложнее отличить, где заканчивается один и начинается другой. Например, задача графического моделирования физического процесса, это задача численной математики, это задача физики или это графическая задача?

В общем, при попытке составить какую-то классификацию задач, мы столкнемся с таким количеством проблем, что невольно придет мысль поискать другой подход.

Попробуем подойти к сложным задачам с другой стороны. Начнем с небольшого, но очень важного замечания. Чтобы мы не изобретали, все упирается в рождение идеи. А идея всегда рождается интуитивно, причем независимо от степени её гениальности и значимости. Происходит это так: вы на некоторое время сосредоточенно размышляете на

заданную тему, и вдруг приходит решение, причем не совсем непонятно откуда. Это называется интуицией.

Все серьезные идеи рождаются интуитивно, что огорчает, так как совершенно не ясно, как интуицией управлять, но с другой стороны, совершенно не подготовленный человек вряд ли сможет создать красивую идею, что вселяет надежду. Ведь если для рождения красивой идеи нужна подготовка, то значит, интуиция где-то в своей основе содержит систему, какой-то метод, а методу можно научиться.

Что такой метод может из себя представлять? Можно ли описать его точно? Конечно нельзя ожидать, что общий метод решения творческих задач будет иметь алгоритмическую ясность. Невозможно себе представить, что такой метод будет описанием последовательности действий. Конечно, многие люди, когда речь идет о методе, представляют некую последовательность инструкций, выполнив которые можно получить верный результат. Но это глубокая, принципиальная ошибка.

Поэтому, мы сразу и навсегда откажемся от идеи разработать такой всеобъемлющий алгоритм. Кроме того, мы решительно откажемся от попыток до конца понять тайну творчества. Это слишком невероятно, чтобы интуиция, творческий инсайт, подлежал исчерпывающему логическому анализу. Это то, что мы не будем делать. А попробуем мы разработать метод, системно использующий интуицию, помогающий удержать направление исследования, способный превратить хаос творчества в осмысленный процесс.

Итак, мы ищем не методы решения задач, а методы организации мыслительной деятельности. Такова наша цель. И чтобы её достичь, не будем строить теорию, а получим умения из практики. Решая задачи и отмечая то, что помогло получить решение, как-то обобщать свои наблюдения и, если получится выводить общие правила.

Сказанное выше не означает ограничения на мыслительную деятельность. Надо просто уяснить, что есть комбинаторное мышление, направленное на построение решения из уже имеющихся кирпичиков, и есть творческое, ставящее целью создание новых знаний.

В этой книге большая часть глав посвящена конкретным задачам, но есть несколько текстов общего значения, как например данная глава. Ниже мы рассмотрим подход, который возможно отражает самую общую идею поиска программисткого решения.

Пошаговое уточнение неопределенностей

Рассмотрим пример несложной задачи. Пусть требуется разработать алгоритм расчета всех простых чисел, не превосходящих заданное число N. Если для решения не требуется особой эффективности, например верхняя граница не слишком велика, то можно воспользоваться самой очевидной идеей: алгоритм решения – это цикл, в теле которого для каждого очередного числа выясняется, простое оно или нет, и если простое, то число печатается как результат.

Для того чтобы принять решение относительно простоты очередного числа, необходимо проверить все числа, могущие быть его делителями, и если хотя бы один делитель есть, то проверяемое очередное не простое, иначе все-таки простое.

Идея вполне понятна, но все же вчитаемся в текст более внимательно, все ли здесь действительно хорошо. Один неясный пункт есть, это фраза требующая проверить все числа могущие быть делителями. Я из своего личного преподавательского опыта знаю, что ученики не достаточно искушенные в задачах математической природы о такие вещи спотыкаются довольно часто. «Как проверить?» это и есть уточняющий вопрос, ответив на который мы сможем записать решение в завершенной форме.

Заметим, что любое исследование начинается с искусства вопроса. Правильно оформленный вопрос, это образно говоря половина ответа.

Рассмотрим более сложный пример. Есть группа людей с вполне определенными, известными симпатиями и антипатиями друг к другу. Необходимо расставить их по рабочим позициям, так чтобы рабочий коллектив оказался психологически наиболее устойчивым. Возможно психология дает какие-то методы решения подобных задач, но мы психологической наукой не владеем. Из соображений здравого смысла ясно, что необходимо организовать перебор всех возможных вариантов, и выбрать из них вариант с наибольшим значением критерия устойчивости.

Это своего рода макроидея, или решение в первом приближении. Здесь сразу видны следующие неопределенности. Во-первых, что значит перебирать варианты. Ясно, что с людьми этого делать не получится, ни в одном языке программирования нет такого термина «люди». Эту неопределенность мы устраняем легко, пронумеровав группу. Тогда перебор вариантов расстановки людей сводится к перебору всех возможных перестановок их номеров. Кстати появившийся термин «перестановка» наводит на мысль поискать какой-нибудь уже известный алгоритм построения всех перестановок. И здесь опять возникает новая неопределенность. Совершенно не факт, что наше понимание термина перестановка совпадает с математическим пониманием. И в этом надо убедиться.

Следующая неопределенность сложнее. Это критерий устойчивости. Ясно, что для каждой полученной перестановки необходимо что-то считать и это что-то должно характеризовать устойчивость построенного коллектива. Мы должны определится с математической формой этого критерия.

Как известно, математика это язык для точной записи знания, поэтому выше не зря появилось упоминание о математической форме записи критерия. Любая работа по устранению неопределенностей, в конечном итоге должна привести, к математически строгой записи.

Таким образом, данная итерация мыслительного процесса должна нам дать формулу расчета критерия и алгоритм, генерирующий перестановки. В общем-то интуитивно ясно, что можно начинать писать код. Но на этапе разработки алгоритма есть еще одна фундаментальная неопределенность – это требование к скорости работы будущей программы или что в значительной степени то же самое – требование к скорости работы алгоритма. Конечно, скорость программы и скорость алгоритма не вполне одно и то же. Но хороший алгоритм можно испортить плохой программой, а вот плохой алгоритм

хорошей программой не исправишь, поэтому вопросы эффективности это прежде всего вопросы алгоритмизации.

В отношении поставленной задачи вопрос скорости очень актуален, мы пришли к необходимости использовать математическую конструкцию перестановки, а как известно для N элементов можно построить N! перестановок. Это очень большая величина и если наш коллектив состоит из более чем 10 человек, чисто переборный вариант решения нас вряд ли устроит и есть смысл подумать о более быстром алгоритме, использующем какие-то более тонкие особенности формулировки задачи или возможность отказа от идеального решения. То есть следующая итерация борьбы с неопределенностью должна дать ответ на вопрос о желаемой эффективности алгоритма.

Таким образом, процесс уточнения неопределенностей заключается в записи решения, так как оно уже понято, выявлении неопределенностей, и формулировки вопросов, ответы на которые устранят неопределенности. Весь процесс можно разбить на ряд итераций, для каждой из которых выполняется очередная запись решения, выявляются очередные неопределенности, формулируются очередные вопросы.

Заметим, что метод ничего не говорит о том, как будет организован поиск ответов на поставленные вопросы, это уже несколько иная проблема. Метод пошагового уточнения, это общая форма организации мыслительного процесса, еще не гарантирующая успешности в частном деле поиска ответов на содержательные вопросы.

Важно заметить, что разбиение мыслительного процесса на фиксированные итерации вещь довольно условная. Конкретная траектория поиска решения зависит от имеющейся системы знаний решателя, от степени развитости интуиции, от банальной удачи. Метод предлагает в общем-то простую вещь. Разбить процесс решения на этапы, в конце каждого решатель должен провести анализ и четко понять, что уже ясно и какие проблемы перед ним стоят.

Таким образом, описываемый процесс уточнения самым прямым образом требует навыков рефлексии (анализа собственного состояния). Вы должны научится постоянному контролю своего интеллектуального состояния. Внутри вашего мыслительного процесса должен постоянно работать маячок, фиксирующий обнаружение проблемы.

Набор проблем кстати величина не постоянная. Появляющиеся решения приближают окончательное решение не в арифметическом смысле, с каждым шагом, все теплее и теплее. Полученное промежуточное решение может поставить новую проблему. Например, в разобранном примере вывод о необходимости алгоритма построения перестановок, ставит перед решателем сложнейшую проблему комбинаторного взрыва.

Усложнение решения процесс объективный. Необходимо понимать, что решатель приступая к задаче, скорее всего не до конца понимает её характер и природу. Процесс решения приводит к более точному пониманию, а следовательно к проявлению проблем, которые существовали объективно, но не были видны решателю.

Еще одно важное замечание о порядке выбора проблем. Предположим, что после очередной итерации мы имеем ряд проблем: A, B, C…. И т.д. Какую из них выбрать для

следующей итерации. Наверное простая логика говорит, что ключевую, то есть такую, которая укажет магистральный путь для решения всей задачи. Наверное это справедливо. Вопрос только в том, что не решив задачу нельзя сказать, какая проблема была магистральной, хотя опыт говорит о том, что интуитивно это как раз почти всегда ясно - ключевая проблема выглядит наиболее сложно.

Однако в этом вопросе есть важный психологический нюанс. Для успешной работы в задачу нужно погрузиться. Погружение, особенно для не вполне опытного исследователя подразумевает хотя бы небольшой, локальный, но успех. Теперь представьте себе ситуацию. Вы приступаете к задаче и четко видите, что нужен ответ на сложный вопрос А и относительно простой ответ В. Вы не сомневаетесь, что ответ на вопрос А будет ключом к решению, ответ на вопрос В имеет чисто технический характер. Простая логика говорит, что необходимо заняться вопросом А. Но быть может лучше решить технический вопрос В!? Вполне возможно, что его решение не очень важно, может быть даже получив ответ на вопрос А, вы поймете, что ответ В был не вполне удачным, но все-таки работа над вопросом В может дать вам именно то необходимое погружение в задачу, без которого не заработает ваш мыслительный аппарат. Это как разминка перед забегом. Она не приносит победы, но готовит мускулатуру к необходимому рывку.

Сказанное означает, что процесс поиска решения не сводится к чисто интеллектуальному действу, набору каких-то логических умозаключений. Решение ищет человек, с определенными психологическими возможностями и особенностями, и это должно быть учтено.

Наконец последнее, но не в последнюю очередь. Говоря о программировании традиционно принято рассказывать о структурном программировании и нисходящем проектировании. Во второй книге нашего курса называемой «Искусство алгоритмизации» этим понятиям посвящена целая глава. Здесь же мы говорим о несколько другом. Обсудим различие. Структурное программирование это парадигма, описывающая технически грамотное построение программы, мы же сейчас обсуждаем не программирование, а процесс поиска решения. Термин «Нисходящее проектирование» несколько ближе. Эта парадигма предписывает разбивать задачу на подзадачи.

Метод же борьбы с неопределенностями предписывает разбивать на части не задачу, а процесс поиска решения, то есть речь по большому счету идет не о статической структуре задачи, а о динамичном мыслительном процессе.

Все последующие главы будут о построении динамичного мыслительного процесса. Описанный выше подход, это основа, фундамент. Далее мы постараемся выделить некий набор эвристических правил, как определить интеллектуальную ситуацию и в какими действиями её изменять в сторону большей определенности. Но это дальше, глава еще не завершена, мы должны обсудить еще некоторые важные вещи

Формализация задачи

Процесс формализации задачи также можно рассматривать, как этап уточнения задачи, но процедура формализации имеет особое значение и особый смысл подготовительного этапа к поиску решения. Проблема любой реальной постановки задачи в неопределенности терминологии, как минимум , а иногда и в упущении некоторых условий. Строгое определение всех объектов участвующих в условии задачи обязательно уберет неточность терминов и возможно поможет выявить пропущенные условия.

Что мы имеем ввиду, когда говорим о возможных неточностях? Проиллюстрируем проблему примерами. Например, в следующей формулировке «Дано множество объектов, найти такую их комбинацию что….» скрывается очень грубая неопределенность делающее условие полностью неясным. Комбинация это термин, не имеющий однозначного смысла. Это могут быть перестановки, это могут быть размещения, сочетания. Процесс перебора комбинаций самым серьезным образом зависит от типа комбинации, поэтому в условии задачи необходимо определить о каком типе комбинации идет речь и переписать условие соответствующим образом, например так: «Дано множество объектов, найти такое их размещение что….». Здесь формулировка уже более строгая, и при этом мы получили значительно больше, нежели просто понимание, мы получили метод решения.

Таким образом формализация выполняет функцию уточнения условия, устраняя избыточность терминологии и создавая предпосылки поиска решения.

Далее. Решение задачи это запись некоторым набором терминов. Условие задачи это также запись некоторым набором терминов. Для того, чтобы работать с заданным условием необходимо терминологическое соответствие между текстом условия и будущим решением. Известно, что любой язык программирования отличается очень небогатым набором выразительных средств, в сравнении с языком естественным, что порождает ситуации очень серьезного смыслового разрыва, в результате которого задача записанная на естественном языке фактически не решаема, если мы не запишем её в терминах языка программирования.

В такую ситуацию постоянно попадают участники программистких олимпиад. Приведем несколько простых примеров.

Пример 1. В классе 30 учеников, между которыми есть как симпатии, так и антипатии, учитель поставил перед собой задачу рассадить учащихся так, чтобы создать в классе максимально комфортную психологическую обстановку.

У нас нет понятия «максимально комфортная психологическая обстановка», поэтому сразу договоримся, что речь идет о допустимой или недопустимой рассадке учеников в классе. Далее заметим, что ученические места в классе представляют собой красивый прямоугольник. Это означает, что неопределенный термин «Рассадка учеников в классе», можно заменить более формальным «заполнение числовыми значениями прямоугольной матрицы». Термин множество учеников можно заменить термином «массив чисел», в котором числа могут играть роль номеров учащихся.

Пример 2. Морской берег имеет сложную конфигурацию……

Не приводим формулировку полностью, она довольно длинная. Сказанного достаточно для иллюстрации нашей проблемы. Здесь прозвучали два естественных термина: «Морской берег», «Сложная конфигурация». Если их заменить языковыми терминами, то может получиться, например, следующая формулировка:

«Дан массив записей содержащих два целочисленных поля…..». Далее термин «целочисленное поле», можно уточнить например так «Поле типа WORD»

В вышеприведенных примерах, вопросы формализации решались достаточно легко. Мы просто подобрали термин алгоритмического языка один в один подходящий для замены термина естественного. Такой метод работает не всегда

Пример 3. Дано число вида A^N …..

Не так важно, что требуется в этой задаче. Угрозы от прямой замены терминов видны невооруженным глазом. Если N – достаточно велико, то рамок целочисленных типов может оказаться недостаточно, а использование действительных типов повлечет за собой неточность в представлении числа, что может оказаться недостаточным. В этой задаче прежде чем решать вопрос о формализации представления данных необходимо подумать проанализировать операции какие должны быть выполнены.

Если например речь идет о выполнении арифметических операций, то может быть разумно представить число в виде массива или связного списка. Если речь идет о исследовании вопросов делимости, то может быть можно вообще обойтись без представления конечного числа, а лучше подумать, как связать свойства основания «A», со свойствами операции возведения в степень. В этом случае каких-то усилий по формализации исходных числовых объектов и не требуется. Привычный целый тип окажется достаточным.

Формализация исходных объектов описанных в условии задачи самым прямым образом связана с потребностями операций, которые предполагается над объектами выполнять.

Пример 4. Лиса на круглом закрытом поле пытается поймать зайца. Скорость зайца равна скорости лисы. Определить оптимальную стратегию для обоих.

Сходу ясно только то, что и лиса должна постоянно находится в движении и заяц может остановится только в том случае, когда стоит лиса. Если начать более глубокий анализ задачи, то мы сразу упираемся в термин «стратегия». Что это такое и как это описать?

Ясно, что стратегия это не объект, это описание действий зайца и лисы, которых кстати легко формализовать парой координат на поле. Со стратегией все не так тривиально. Стратегию поведения зайца можно описать кривой линией при условии что траектория лисы известна. Но траектория лисы, по соображениям здравого смысла зависит от траектории зайца. А это вроде бы означает невозможность точного описания. Здесь есть два выхода из положения.

Выход первый. Описание ситуации в некоторой небольшой пространственной или временной окрестности.

Таким способом часто решают свои проблемы физики и математики. Например, зная значение производной от функции в точке, можно интегрированием восстановить внешний вид функции. Зная локальные характеристики газа в небольшом объеме можно представить его поведение в большом сосуде. Если мы знаем, скорость тела на малом участке пути и знаем, что никакие силы на него не действуют, то мы знаем, как оно будет двигаться дальше. Это намек на то, что зная общее правило для принятия решение лисой и зайцем в конкретный момент времени мы можем выстроить и траекторию. Вполне возможно, что у неё окажется какой-то очень простой вид. В этом случае решение задачи сводится к поиску такого правила.

Выход второй. Все же возможно решение макрозадачи.

В условии было сказано, что для обоих необходимо построить оптимальную стратегию. Мы уже решили, что термин «стратегия» удачно заменяется термином «траектория». Далее, можно вполне обоснованно выдвинуть гипотезу, что при заданной форме поля существует только одна оптимальная стратегия, а следовательно и только одна траектория. Справедливость этой гипотезы будет означать, что исходную задачу можно переформулировать так

«Построить для преследуемого и преследователя такие траектории, что отступление от оптимума со стороны преследователя будет означать увеличение расстояния, а отступление со стороны преследуемого будет означать сокращение расстояния.»

В некотором смысле можно сказать, что формализация задачи это ответ на вопрос, что же на самом деле требуется получить в решении и как это должно выглядеть.

Еще можно сказать, что формализация задачи это запись условия и процесса решения в виде, не допускающем многозначного толкования, что в свою очередь дает еще одно определение формализации, как запись, на формальном языке. В таком коротком параграфе дано сразу несколько пониманий термина «формализация», но по большому счету, таким образом всего лишь показаны разные стороны одной вещи.

Решение как построение цикла Дейкстры

А сейчас позвольте сформулировать гипотезу относительно формы любого программисткого решения. Сразу хочу заметить, что это не более чем моя личная гипотеза, поэтому не судите слишком строго, а звучит она так:

Решение алгоритмически чистой задачи сводимо к построению цикла Дейкстры.

Здесь в соответствии с нашей же технологией рассуждений, необходимо убрать две неопределенности. Во-первых, надо пояснить, что такое алгоритмически чистая задача, и во-вторых, возможно не все знают, что такое цикл Дейкстры.

Цикл Дейкстры

Э. Дейкстра в теории систематического вывода императивных программ ввел многоветочную форму цикла WHILE представляющую собой описание циклического процесса управляемого несколькими условными операторами. В языке Компонентный Паскаль цикл Дейкстры можно реализовать циклом безусловным циклом LOOP.

Пример:

```
LOOP
  IF условие THEN EXIT
  {ELSIF условие THEN группа операторов}
  …….
  END;
END;
```

Фигурные скобки ограничивающие условный оператор означают возможность многократного повторения этой части. Цикл Дейкстры позволяет исключить необходимость вложенных циклов.

Пример:

Структура с вложенными циклами	Идентичная структура с циклом Дейкстры

FOR x:=1 TO N DO
  FOR y:=1 TO M DO
  Группа операторов
  END;
END;
	x:=1;y:=1;
	LOOP
	  IF x>N THEN EXIT
	  ELSIF y>M THEN
	  x:=x+1; y:=1;
	  ELSE
	  Группа операторов
	  END;
	END;

Указанное свойство цикла Дейкстры позволяет утверждать, что имеет место следующее соответствие: Один процесс (даже сложный) = один цикл Дейкстры

Алгоритмически чистая задача

Назовем таковой задачу, для которой можно описать исходный набор данных и конечный результат, также в виде некоторого набора данных. Заметим сразу, что системы управления чем либо, системы обработки баз данных, под данное определение не подходят, там важен процесс, начало которого по большому счету исходными данными не определяется, там скорее важна временная точка в которой программа начала работать. Нет и завершения, есть только некоторые промежуточные состояния.

Под наше определения попадают расчетные задачи наверное любого характера, а также задачи цель которых ответить на вопрос справедливо некоторое утверждение или нет.

То есть, алгоритмически чистая задача реализуется процессом имеющим начало и завершение, процесс инициируется некоторым набором данных и завершается некоторым набором. Любой процесс, как мы знаем состоит из ряда итераций, возможно имеющих сложную структуру, но выше мы выяснили, что сложная структура операций сводится к одному циклу Дейкстры, что собственно и требовалось показать.

Конечно, данный текст нельзя воспринимать, как строгое доказательство, но такой цели и не ставилось, здесь изложена не более чем гипотеза, надеемся, что правдоподобная.

В заключение

Таким образом, получив формулировку задачи первое, что мы должны сделать, это привести её в соответствии с имеющимся интеллектуальным инструментарием, записать в общепринятых математических терминами, терминах языка программирования, как результат процесса формализации точно понять, что от нас требуется на самом деле.

Затем решатель начинает процесс снятия неопределенностей в своем знании о задаче. Он исследует условие, выясняет, что ему в задаче ясно, что нет. На основе полученных неопределенностей формулирует вопросы становящиеся отправными пунктами последующего исследования. И так до получения решение в окончательном виде.

Конечно все сказанное выше это не более чем каркас на который навешиваются более тонкие и более содержательные методы и приемы. Их исследование и будет предметом всех последующих глав этой книги.