Как решаются задачи. (По впечатлениям от Летней школы юных программистов в г.Новосибирске).

Во многих школах появилась кое-какая вычислительная техника, а вместе с ней и практика на ЭВМ. И вот, учащиеся, почувствовавшие к управлению компьютерным ироm, собрались на 12-ой Летней школе юных программистов в молодёжном Туристком центре "Сибиряк" под городом Новосибирском. Как обычно работа школы была нацелена на закрепление практических навыков работы на ЭВМ и расширение кругозора в области программирования. Неожиданно обнаружилось, что многие, вкусившие от БЭЙСИК(а), искренне считают свой программисткий багаж вполне достаточным. Внимание уделяют лишь техническим деталям программирования на тех ЭВМ, которые доступны по месту жительства. Как это не парадоксально звучит, но такая приземлённая нелюбознательность является симптомом недобросовестного отношения к любимому делу.

Избегая прямых аналогий, позволим себе маленькое отступление. Известный этолог (этология--наука о поведении животных) Конрад Лоренц в своей книге "Кольцо царя Соломона" описывает поведение небольшого зверька --водяной землеройки--куторы. Этот зверёк хорошо ориентируется в воде, но на суше чувствует себя очень неуверенно. Если кутора успешно пробралась к приманке или к другой цели очень извилистым путём, обходя препятствия, то потом, когда уже не будет препятствий, она всё равно их будет "обходить". Она не догадывается проверить, не исчезли ли препятствия или поискать более близкий путь. Свой путь к цели она, похоже, запоминает в виде программы движения, состоящей из последовательности команд "вперёд", "направо", "налево". Лишь если путь станет невыполнимым из-за новых препятствий, кутора может случайно найти более короткий путь. (Кстати ученик привыкший исполнять инструкции не вдаваясь в их смысл и происхождение, то есть исполнять ритуал, не способен даже на это. Он просто сдасться перед возникшей проблемой(поздняя вставка)). Животные, более развитые чем кутора, обычно проверяют не исчезли ли препятствия и быстро находят более короткий путь. Наверное они по программе раз пойденного пути могут "представить" поверхность на которой этот путь лежит, и даже как бы взглянуть на эту поверхность сверху, чтобы найти оптимальное направление движения. Можно сказать что в этом случае задача поиска пути решается обобщением --переходом в пространство большей размерности, представление о котором формируется под воздействием программы движения по допустимому пути. Большая размерность поискового пространства создаёт предпосылки (условия) для поиска более короткого пути, но на такой качественный скачок у куторы не хватает потенциала.

Склонность к привычному поведению в определённой степени свойственна и людям, которые меняют его лишь под давлением обстоятельств и, подчас, очень неохотно. Необходимо сознательно развивать в себе готовность к нестандартным решениям, отходу (отказу) от привычного поведения, систематически обращаясь к задачам, решения которых традиционными методами невозможно (к задачам, для решения которых нет готовых инструкций). Необходимо расширять спектр ассоциаций, аналогий, обеспечивая взаимодействие различных знаний и средств. Для этого, по крйней мере, не нужно

отгораживаться от информации несущей новое качество, новый уровень рофессиональной подготовленности. *Ведь именно новые ассоциативные связи, возникающие и закрепляющиеся в мозгу, способствуют повышению размерности поискового пространства.*

Удивляет отношение многих ребят, весьма неплохо владеющих техникой и языками программирования, к такой фундаментальной области знания, как математика. Оно либо пренебрежительное, либо в лучшем случае уважительное, но, предполагающее независимое от нужд программирования, существование.

Это заблуждение характерно для школьников и многих профессионалов программистов, их обучающих. Его нельзя устранить оставаясь в рамках традиционно школьного образования даже в его нынешнем варианте, осовремененном курсом информатики. Особенно отчётливо этот недостаток выявился при проведении в Летней Школе олимпиады по программированию, которая показала, что очень немногие умеют анализировать анализировать задачу, качественно проводить алгоритмизацию, стремясь к наиболее эффективным алгоритмам, доказывать их полное соответствие поставленной цели. Только двое ребят представили грамотные и достаточно обоснованные алгоритмы решения поставленных задач и то, что они призёры всесоюзных олимпиад по математике, то есть имеют достаточную подготовку в решении нестандартных задач, лишь подтверждает сказанное выше.

Среди представленных решений встречались программы, про которые невозможно было понять, что они делают и работают ли они вообще. При этом необходимость представить алгоритм решения задачи выполнялась в лучшем случае как малообоснованная уступка требованиям организаторов и при том в форме весьма небрежной и отвлечённой от представленной программы.

Для иллюстрации приведенных соображений рассмотрим одну из задач, предложенных на этой олимпиаде и покажем несколько основных типов решений на алгоритмическом языке, в порядке приближения к самому эффективному варианту.

Задача.

Для заданного натурального числа n сгенерировать все различные треугольники с целочисленными сторонами и периметром равным *n*.

Сразу возникает вопрос о способе представления информации задающей треугольник. Задать (описать) треугольник можно по разному. В данном случае напрашивается следующий способ--задавать треугольники тройками натуральных чисел a,b,c удовлетворяющих неравенствам треугольника, то есть системе неравенств

$$\begin{cases} a+b>c \\ b+c>a \\ c+a>b \end{cases}$$

Если добавить исходное требование $a+b+c=n$, где n --заданное натуральное число, то в принципе уже на этом этапе средствами алгоритмического языка можно написать алгоритм для нахождения всех искомых треугольников. Впрочем, используя любой язык программирования (Бэйсик, Паскаль, Рапиру и другие), можно сразу писать программу для решения этой задачи. Что собственно и сделало большинство участников олимпиады (сочтя данную задачу удивительно лёгкой), выполнив, при этом, практически вырожденный этап алгоритмизации в уме. Ведь ясно, что достаточно просто организовать перебор троек натуральных чисел a,b,c, где каждое независимо пробегает все значения от *1* до *n-2* с шагом 1 и оставлять только те тройки, которые удовлетворяют одновременно четырём условиям: $a+b+c=n$, $a+b > c$, $b+c > a$,$c+a > b$.

Имея в виду такой алгоритм, большинство участников сочло задачу тривиальной и не помышляли об ином подходе к её решению, тем более, что такой способ, реализованный на ЭВМ действительно позволяет получить решение поставленной задачи.

Поскольку численный эксперимент на ЭВМ обычно ограничивается небольшим исходным n, поводов для беспокойства и неудовлетворённости не возникало. Следующий шаг, который можно и нужно сделать и, который сделали некоторые участники олимпиады, заключался в том, чтобы заставить программу выдавать только те тройки чисел, которые задают различные треугольники. Ведь в описанном выше алгоритме вложенные циклы, реализующие все возможные тройки, естественным образом их упорядочивают, то есть выдают на выходе упорядоченные тройки натуральных чисел в то время как треугольник задаётся неупорядоченной тройкой чисел $\{a,b,c\}$.

Цикл, генерируя упорядоченную тройку чисел (a,b,c), генерирует ещё пять троек, являющихся перестановками этой тройки. И все эти 6 троек по сути задают один и тот же треугольник. Описанный выше недостаток устраняется простым и понятным способом--добавлением дополнительного условия $a \geq b \geq c$. Одни участники олимпиады механически добавили это условие к перечню условий приведенного выше алгоритма, а другие замечали, что система неравенств-условий

$$\begin{cases} c < a + b \\ a < b + c \\ b < c + a \\ c \leq b \leq a \end{cases}$$

эквивалентна системе

$$\begin{cases} b < c + a \\ c \leq b \leq a \end{cases}$$

(так как все числа a,b,c натуральные, то оставшиеся два неравенства $a+b>c$ и $c+a>b$ являются следствием неравеннства $a\geq b\geq c$) и предлагали вариант того же алгоритма, где условия отбора имели вид

$$\begin{cases} a+b+c=n \\ c\leq b\leq a \\ a<b+c \end{cases}.$$

Внимательный и вдумчивый читатель уже наверное давно недоумевает, зачем нужны эти банальности. Ведь ясно же, что $c=n-a-b$ и поэтому перебор значений c не нужен и, что верхняя граница для a,b,c никак не может быть равна $n-2$ уже для $n=5$.

Более того из $a<b+c$ и $a+b+c=n$ следует $2a<a+b+c$, то есть $2a<n$ Отсюда, так как a и n натуральные числа,

$$2a\leq n-1 \Leftrightarrow a\leq \left[\frac{n-1}{2}\right].$$

Действительно, учёт этих соображений позволяет рассмотреть ещё один вариант алгоритма, построенного более экономно по сравнению с предыдущим и, который, с различными вариациями представила достаточно многочисленная группа участников олимпиады.

Приведём соответствующий этому алгоритму операционный блок на алгоритмическом языке:

```
a := 1; b := 1
пока a ≤ (n − 1)/2
  нц
    пока b ≤ a
      нц
        c := n − a − b
        если c ≤ b
          то вывод a, b, c
        b := b + 1
      кц
    a := a + 1
  кц
всё
```

Но и этот алгоритм далёк от совершенства, поскольку генерирует двойки чисел (a,b) такие, которые отбрасываются на этапе проверки условия $c \le b$ (например: при $n=5$ вложенными циклами генерируются значения a и b равные единице и соответствующее им $c=5-1-1=3>b$).

Между первым алгоритмом и этим лежит целый спектр решений, в которых лучше или хуже работают условия, сокращающие неэффективный перебор. Но все эти алгоритмы, так или иначе, работают как решето--одни тройки отсеивают, а другие оставляют.

В данной задаче, при наличии эффективного(бесперебороного) алгоритма, такие решения нельзя назвать приемлемыми. Более того, многие вступившие на путь сознательного уменьшения количества состояний алгоритма, приводящих к отсеиванию неприемлемых троек не пошли дальше, приведеннного выше, алгоритма. Отождествляя исходные неравенства-условия с некоторым набором следствий из них, они не позаботились о соблюдении равносильности преобразований и получали программы, которые были ошибочными, так как пропускали в виде решений тройки чисел, которые треугольника не задавали.

Грамотное решение задачи состоит в направленном равносильном преобразовании исходных неравенств-условий, что по сути есть техника математическая и вполне доступная школьнику(точнее сказать весма желательная для него).

Весь процесс решения задачи можно представить примерно так (возможны варианты):

$$(1)\quad \begin{cases} a+b+c=n \\ 1 \le c \le b \le a \\ a < b+c \\ b < c+a \\ c < a+b \end{cases} \overset{a,b,c\in\mathbb{N}}{\Leftrightarrow} \begin{cases} c=n-a-b \\ 1 \le c \le b \le a \\ a+1 \le b+c \end{cases} \Leftrightarrow \begin{cases} c=n-a-b \\ b \le a \\ 1 \le n-a-b \le b \\ a+1 \le b+n-a-b \end{cases} \Leftrightarrow$$

$$\begin{cases} c=n-a-b \\ b \le a \\ b \le n-a-1 \text{ and } n-a \le 2b \\ 2a \le n-1 \end{cases} \Leftrightarrow \begin{cases} c=n-a-b \\ b \le a \le \min\{a, n-a-1\} \\ n-a \le 2b \\ 2a \le n-1 \end{cases}.$$

Так как $n-a \le 2b \Leftrightarrow \left[\frac{n-a+1}{2}\right] \le b$, $2a \le n-1 \Leftrightarrow a \le \left[\frac{n-1}{2}\right]$ для целых a и $b \in \mathbb{N}$

и $\min\{a, n-a-1\} = a$ для $a \le \left[\frac{n-1}{2}\right]$ мы получим (1) $\Leftrightarrow \begin{cases} c=n-a-b \\ \left[\frac{n-a+1}{2}\right] \le b \le a \\ a \le \left[\frac{n-1}{2}\right] \end{cases}.$

И наконец, из неравенства

$$\left[\frac{n-a+1}{2}\right] \le b \le a$$

следует (как условие его непротиворечивости)

неравенство $\left[\frac{n-a+1}{2}\right] \le a$ которое, так как a целое, равносильно неравенству

$\frac{n-a+1}{2} < a+1 \Leftrightarrow n-a+1 < 2a+2 \Leftrightarrow n-a+2 \le 2a+2 \Leftrightarrow n \le 3a \Leftrightarrow \left[\frac{n+2}{3}\right] \le a.$

Таким образом, учитывая что $\left[\frac{n+2}{3}\right] \ge 1$ для натурального n мы получим систему

(**2**)
$$\begin{cases} \left[\frac{n+2}{3}\right] \le a \le \left[\frac{n-1}{2}\right] \\ \left[\frac{n-a+1}{2}\right] \le b \le a \\ c = n-a-b \end{cases},$$

равносильную исходной системе **(1)** и эффективно задающую все искомые тройки натуральных чисел (a,b,c).

На алгоритмическом языке ей соответствует операционный фрагмент

```
a := [(n+2)/3]
пока a ≤ [(n-1)/2]
  нц
    b := [(n-a+1)/2]
    пока b ≤ a
        нц
          c := n-a-b
          вывод a,b,c
        b := b+1
      кц
    a := a+1
  кц
всё
```

который генерирует все различные треугольники с целочисленными сторонами и периметром равным n и при этом ни одна из генерируемых троек не отбрасывается.

Таким образом система **(2)** по сути является решением поставленной задачи. Она наиболее приспособлена к целям алгоритмизации в том смысле, что иммет структуру удобную для реализации в виде вложенных циклов и даёт возможность написать

наиболее эффективную программу, в которой обьём выполняемых операций оптимально согласован с обьёмом выдаваемой информации(в том смысле, что выдаётся столько троек сколько состояний генерирует вложенные циклы).

Но было бы неверным сказать, что алгоритмизация закончена, поскольку осталось ответить ещё на один вопрос о множестве допустимых значений n, то есть найти то множество значений n для которых эта задача разрешима. Последняя система неравенств позволяет легко ответить и на этот вопрос.

Ответ очевиден -- задача разрешима для таких натуральных n, для которых выполняется неравенство

$$\left[\frac{n+2}{3}\right] \le \left[\frac{n-1}{2}\right].$$

Предлагается читателю самостоятельно решить это неравенство в натуральных n и убедиться, что множество его решений $\setminus\{1,2,4\}$.

В заключении заметим , что неудовлетворённость пренебрежительным отношением "фанатирующих" юных программистов к процессу алгоритмизации и, вообще, к использованию математики в их собственной работе, была столь острой, что у одного из авторов этой статьи это вылилось в своего рода воззвание, которое в стихотворной форме было вывешено на доске обьявлений. Им же мы, завершая статью, обращаемся ко всем читателям журнала Квант, увлечённых программированием и желающим достичь определённой культуры в этой области (и в мышлении вообще).

Неужто среди вас

Приверженцев Агатов и Ямах

Не встретиться таких, чьё сердце отзовётся

Когда с присущей красотой и блеском

На программистских на нехоженых тропах

Вам математика свои протянет руки

И вы невольные виновники разлуки

Рванётесь к ней навстречу

И с нею обретёте новый мир.

Нет не померкнет прежний ваш кумир.

К нему вернётесь вы стократно усилённый

И прежней слепотой своей безмерно изумлённый

Постигнуть сможете другие измеренья

Источники для творчества и вдохновенья.

Городняя Л.В, Альт А.М. (Новосибирск, Одесса) 1988 г.

Приложение.

Решение неравенства $\left[\frac{n+2}{3}\right] \le \left[\frac{n-1}{2}\right]$ в натуральных числах.

Так как

$$\frac{n+2}{3} \le \frac{n-1}{2} \Leftrightarrow 2n+4 \le 3n-3 \Leftrightarrow 7$$

и

$$\frac{n+2}{3} \le \frac{n-1}{2} \Rightarrow \left[\frac{n+2}{3}\right] \le \left[\frac{n-1}{2}\right]$$

то для всех $7 \le n$ неравенство $\left[\frac{n+2}{3}\right] \le \left[\frac{n-1}{2}\right]$ выполняется. Нетрудно прямой подстановкой убедится в том, что из оставшихся шести натуральных чисел неравенству $\left[\frac{n+2}{3}\right] \le \left[\frac{n-1}{2}\right]$ не удовлетворяют только $n=1,2,4$.

Таким образом $\left[\frac{n+2}{3}\right] \le \left[\frac{n-1}{2}\right] \Leftrightarrow n \in \mathbb{N} \setminus \{1,2,4\}.$