Задания заключительного тура

Всероссийского конкурса научных работ школьников «Юниор» 2014-2015 учебного года

секция информатики, 9 класс

Задача A. «Опечатка»

Максимальное время работы на одном тесте: *2 секунды*
Максимальный объем используемой памяти: *64 мегабайта*

Начинающий хакер Петя очень занят разработкой плана по взлому Самого Главного Сервера Интернета. Петя давно бы уже разработал этот план, если бы ему не приходилось постоянно отвлекаться на всякие глупости – на то, чтобы убраться в своей комнате, почистить зубы или – вот как сейчас – на школьные домашние задания.

Сегодняшнее домашнее задание Пети по информатике состоит в том, чтобы написать программу, проверяющую, возможно ли для трех введенных целых чисел a, b, c построить невырожденный треугольник с соответствующими длинами сторон. Вспомнив, что учитель информатики говорил что-то про использование неравенства треугольника, Петя быстренько написал свой вариант программы для решения этой задачи. К сожалению, программа Пети возвращает неправильный ответ для второго теста из условия задачи. Поскольку Пете требуется незамедлительно вернуться к планированию взлома Самого Главного Сервера, он просит Вас доделать вместо него его домашнее задание.

Ниже приведен листинг программы, написанной Петей.

```
Program Triangle;
var
  a,b,c:integer;
begin
  readln(a,b,c);
  if ((a-b+c) > 0) and
      ((a-c+b) > 0) and
      ((a-b-c) > 0)
  then
    writeln('YES')
  else
    writeln('NO');
end.
```

Формат входных данных

В единственной строке входных данных содержатся три целых числа a,b,c, разделенных пробелами: $0 < a,b,c \leq 1000$

Формат результата

В единственную строку результата вывести "YES" (без кавычек, заглавными буквами), если из отрезков с длинами a,b,c можно составить невырожденный треугольник, и "NO" (без кавычек, заглавными буквами) в противном случае.

Пример входных данных и результата:

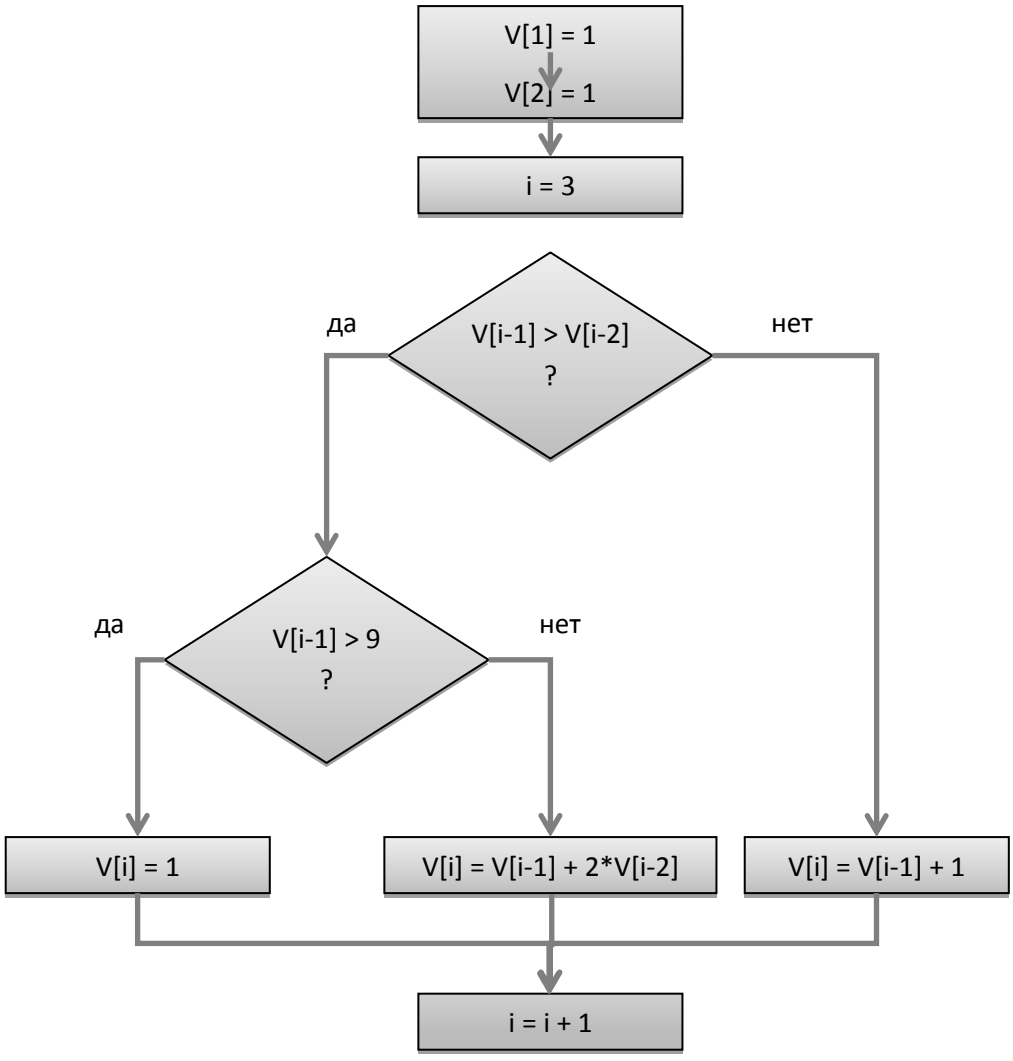
standard input	standard output
3 1 5	NO
4 2 3	YES

Задача B. «Числа Васиаччи»

Максимальное время работы на одном тесте: *2 секунды*
Максимальный объем используемой памяти: *64 мегабайта*

Начинающему хакеру Васе не даёт покоя слава Великого Математика древности Фибоначчи, про которого вчера рассказывал школьный учитель. Вася решил, что стать таким же Великим Математиком– гораздо круче и почётнее, чем Великим Хакером. Тем более, что в последнее время у Васи возникли сложности с разработкой его собственного плана по взлому Самого Главного Сервера (только не рассказывайте об этом Пете!)

Таким образом, Вася твердо решил стать Великим Математиком. Для этого Васе нужна его собственная последовательность чисел, названная в его честь. Чтобы известность и слава Великого Математика пришли как можно быстрее, Вася собирается сделать свою последовательность как можно хитрее. Всю вчерашнюю ночь он не мог заснуть и, в конце концов, нарисовал следующую схему, по которой должны формироваться числа его последовательности:



Теперь Вася хочет научиться узнавать значение произвольного элемента своей последовательности V_n по заданному номеру элемента n. К несчастью, гораздо больше Вася хочет немного поспать. Поэтому, взяв с Вас обещание не раскрывать секрет его последовательности, Вася показал Вам свою схему и попросил написать программу, которая позволит по заданному номеру элемента n вычислять значение элемента V[n] Васиной последовательности.

Не забудьте, что, хотя Вася и нарисовал в своей блок-схеме бесконечный цикл, Ваша программа должна вернуть ответ за ограниченное время.

Формат входных данных

В единственной строке входных данных содержится единственное целое число n, такое что: $1 \le n \le 100000$

Формат результата

В единственную строку результата вывести единственное целое число – искомое значение n-го элемента последовательности.

Пример входных данных и результата:

standard input	standard output
1	1
5	8

Задача C. «Эврика!»

Максимальное время работы на одном тесте: *2 секунды*
Максимальный объем используемой памяти: *64 мегабайта*

Вот оно! И как только Петя не догадался об этом раньше?! Все дело – в прямоугольных треугольниках!

Эта гениальная мысль пришла в голову Пете совсем внезапно, когда уже казалось, что его план по взлому Самого Главного Сервера снова зашел в тупик. И вдруг – вот оно, решение! Действительно, попробуйте вспомнить – где в дикой природе вы встречали ровные прямые углы? Нет, отдельные частные случаи в расчет брать не будем – наверняка, если измерить очень внимательно, то окажется, что углы там тоже не совсем прямые. А теперь посмотрите вокруг, на то, что человек создал искусственно. Видите? Сплошные прямые углы! Один за другим! Везде, все состоит из прямых углов!

Петя уверен, что ему удалось обнаружить Фундаментальный Критерий того, что произвольно взятый объект является созданным руками человека. Петя пока не знает точно, как именно применить этот Критерий для взлома Самого Главного Сервера – но он уверен, что после такого Фундаментального Открытия решение всех остальных вопросов будет уже делом техники. Поэтому сейчас Петя решил взять небольшой отдых, чтобы сходить проведать Васю и выяснить, как продвигается его план, а заодно и невзначай сообщить Васе о своих собственных успехах. Вас же, чтобы не терять время впустую, Петя попросил написать программу, реализующую на практике проверку его Фундаментального Критерия. Программа должна по заданным координатам трех точек на плоскости определить, являются ли данные точки вершинами прямоугольного треугольника, и вычислить длину его гипотенузы.

Формат входных данных

В трех строках входных данных содержатся по два целых числа: координаты X,Y трех вершин соответственно: $-100 \le Xi, Yi \le 100$. Гарантируется, что заданные точки попарно различны и не лежат на одной прямой.

Формат результата

В единственную строку результата вывести единственное вещественное число – длину гипотенузы, с точностью не менее трех значащих цифр после запятой, в случае если заданные вершины образуют прямоугольный треугольник. Если вершины не образуют прямоугольный треугольник, вывести в качестве результата число -1.

Пример входных данных и результата:

standard input	standard output

0 4 3 0 3 4	5.000
2 -1 -1 1 -1 -2	-1

Задача E. «Салочки»

Максимальное время работы на одном тесте: *2 секунды*
Максимальный объем используемой памяти: *64 мегабайта*

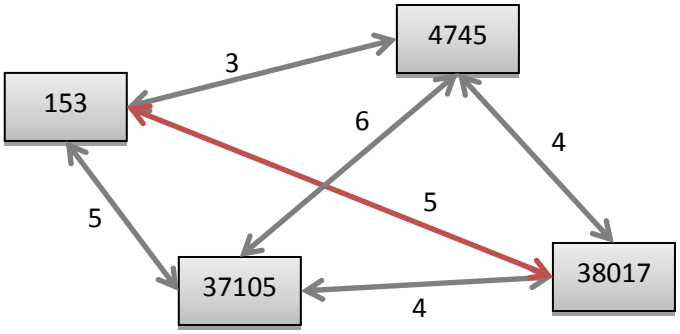
Алиса (ALICE - Artificial Lively Intelligent Creature Entity) и Боб (BOb - Binary Object) играют в салочки. Поле, на котором они играют, представляет собой множество регистров оперативной памяти, соединенных переходами. Каждый регистр имеет номер от 0 до 65535. Длина перехода между двумя регистрами равна количеству различающихся бит в двоичных представлениях номеров этих регистров. Например, расстояние между регистрами #153 и #38017 равно 5:

153_{10}= **0**00**0** 0**0**00 100**1** **1**001$_2$
38017_{10}= **1**00**1** 0**1**00 100**0** **0**001$_2$

Алиса знает, что Боб умеет перемещаться по памяти на долю такта быстрее, чем она сама. Поэтому единственный способ выиграть для неё состоит в том, чтобы заблокировать прямой переход между регистрами после её собственного перемещения. Теперь Алиса хочет знать, существует ли для данной пары регистров на поле какой-либо другой путь, суммарная длина которого не превышает длину прямого перемещения.

Например, пусть поле для игры состоит из 4-х регистров: #153, #4745, #37105, #38017

153_{10}= 0000 0000 1001 1001$_2$
4745_{10}=0001 0010 1000 1001$_2$
37105_{10} = 1001 0000 1111 0001$_2$
38017_{10} = 1001 0100 1000 0001$_2$



Из рисунка видно, что, если заблокировать прямой переход между регистрами #153 и #38017, то длина кратчайшего пути между этими регистрами (#153 => #4745 => #38017) составит 7, то есть, больше, чем длина прямого перемещения (5).

Формат входных данных

В первой строке входных данных содержится единственное целое число N: $3 \le N \le 64$ – количество регистров памяти, из которых состоит поле. В следующих N строках входных данных указывается по одному целому числу R[i] – номер соответствующего регистра памяти, $0 \le R[i] \le 65535$. Гарантируется, что все номера регистров различны. В последней строке входного файла указываются через пробел два различных числа A,B – номера регистров, между которыми перемещается Алиса. Гарантируется, что регистры A и B принадлежат игровому полю.

Формат результата

Если существует еще хотя бы один путь между регистрами A и B, суммарная длина которого равна длине прямого перехода, в единственную строку результата вывести одно целое число – эту длину. Если пути, удовлетворяющего условию, не существует, вывести в качестве результата -1.

Пример входных данных и результата:

standard input	standard output
4	-1

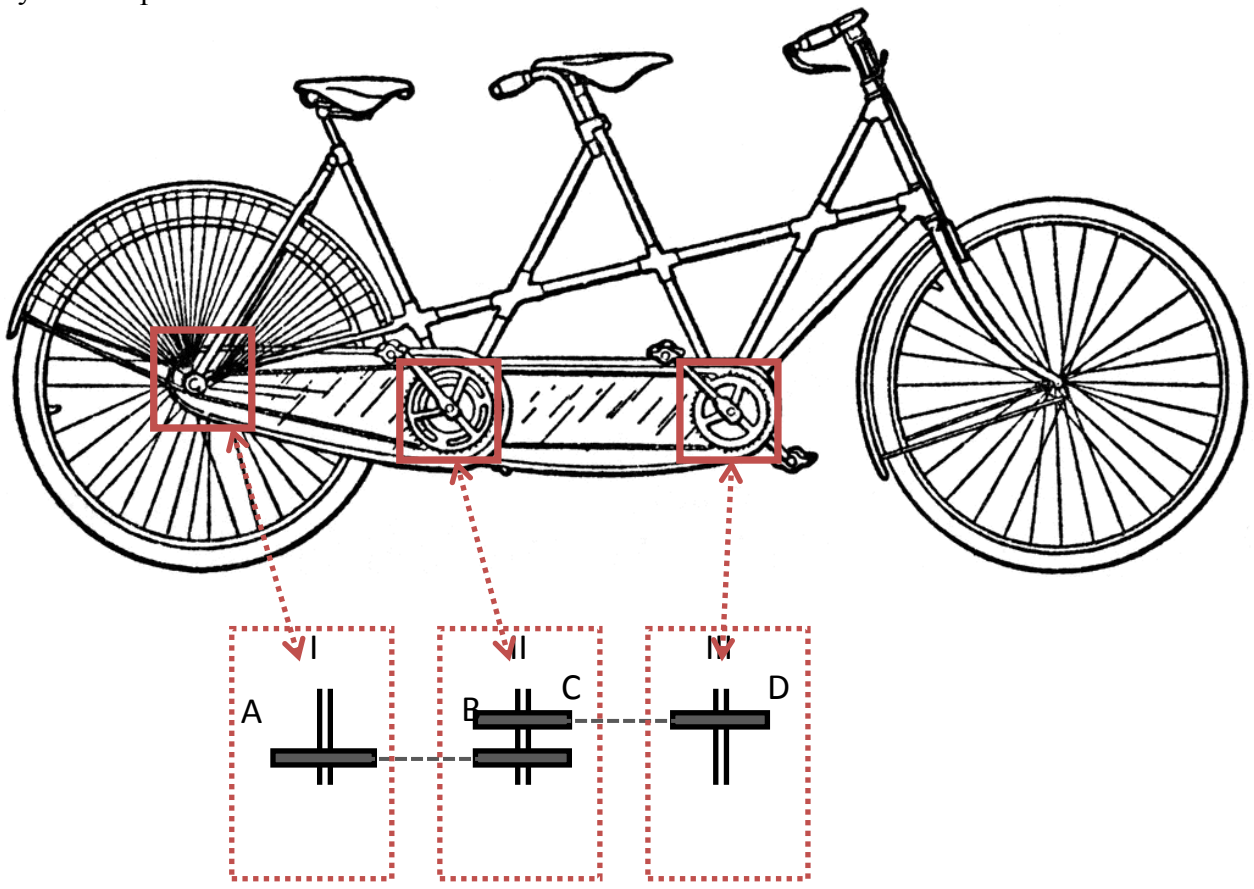
4745 153 37105 38017 153 38017	

Задача F. «Тандем»

Максимальное время работы на одном тесте: *2 секунды*
Максимальный объем используемой памяти: *64 мегабайта*

Альберт и Альфред снова изобретают велосипед. На этот раз, правда, речь идет о велосипеде-тандеме. Их не устраивает классическая конструкция такого велосипеда, когда оба велосипедиста должны крутить педали с одинаковой скоростью. Альфред любит очень точную настройку передаточных чисел, такую, чтобы заднее колесо велосипеда крутилось ровно вдвое чаще, чем частота вращения педалей. Также Альфред любит сидеть за рулем, на переднем сиденье тандема. Альберт же предпочитает более спокойную езду, когда можно спокойно любоваться окрестными пейзажами, и не нужно крутить педали очень часто.

Чтобы реализовать эту идею, братья хотят использовать звездочки с разным количеством зубьев на разных осях:



Теперь братья хотят выяснить, могут ли они собрать нужную им конфигурацию, используя имеющиеся у них в наличии шестеренки.

Формат входных данных

В первой строке входных данных содержится единственное целое число N: $4 \le N \le 50$ – количество шестеренок, которые есть у братьев. В следующих N строках входного файла указывается по одному целому числу T_i: $3 \le T_i \le 1000000$ – количество зубчиков на соответствующей шестеренке.

Формат результата

Если возможно из имеющихся шестеренок выбрать четыре таких, что выполняются следующие условия:

1) Угловая скорость вращения оси I ровно вдвое выше, чем угловая скорость вращения оси III
2) Угловая скорость вращения оси II меньше, чем угловая скорость вращения оси III

то вывести в качестве в качестве результата в единственной строке четыре целых числа, разделенных пробелами: номера четырех шестеренок в порядке A, B, C, D (согласно маркировке на рисунке). Если вариантов решения задачи несколько, вывести любой из них, удовлетворяющий условию.
Если решения, удовлетворяющего условию, не существует, вывести в качестве результата единственное число -1.

Пример входных данных и результата:

standard input	**standard output**
5 4 11 77 44 8	2 4 5 1
4 999999 500000 899999 900000	-1

Ответы и решения

Задача A. «Опечатка»

Возможность для трех заданных чисел a, b, c построить невырожденный треугольник с соответствующими длинами сторон проверяется при помощи системы неравенств треугольника, выписанных для каждой из трех его сторон. Для заданных чисел должны выполняться следующие условия:

$a < b + c$
$b < c + a$
$c < a + b$

Несложно заметить, что в приведенном листинге программы Пети одно из этих неравенств записано неверно:

```
if ((a-b+c) > 0) and
     ((a-c+b) > 0) and
     ((a-b-c) > 0)
then ...
```

Правильный вариант этого фрагмента кода может выглядеть, например, следующим образом:

```
if ((a-b+c) > 0) and
     ((a-c+b) > 0) and
     ((a-b-c) < 0)
then ...
```

Задача B. «Числа Васиаччи»

По приведенной блок-схеме, согласно которой Вася формирует числа своей последовательности, видно, что значение каждого следующего числа V[i] последовательности зависит не более чем от двух предыдущих: V[i-1], V[i-2]. С учетом того, что формулы определения следующего числа последовательности очень просты, а ограничение на максимальный номер числа, которое нужно посчитать, мало ($n \le 100000$), можно реализовать вычисление нужно числа через обычный цикл, внутри которого аккуратно переписать на соответствующем языке программирования логику, описанную в блок-схеме. Отдельно нужно заметить, что ветка блок-схемы «если V[i-1] > 9 то V[i]=1» ограничивает рост абсолютных значений элементов последовательности, поэтому даже элементы с большими номерами будут умещаться в стандартные целочисленные типы.

Впрочем, решать данную задачу можно и без использования цикла и вычисления всех элементов последовательности. Та же самая ветка блок-схемы, ограничивающая рост элементов: «если V[i-1] > 9 то V[i]=1» обеспечивает, что последовательность является циклической, причем период последовательности значительно меньше, чем порядок максимально возможного значения n: $n \le 100000$.

Для того чтобы выделить период этой последовательности, достаточно построить несколько первых её элементов:

1, **1, 2, 4, 8, 16**, 1, 2, 4, 8, 16, 1, 2, 4, …

Начиная со второго элемента, последовательность является циклической с периодом в 5 элементов, а сам циклический фрагмент состоит из значений 1, 2, 4, 8, 16. Выявив данную закономерность в структуре последовательности, значение любого элемента последовательности становится легко определить аналитически по номеру элемента.

Задача C. «Эврика!»

Поскольку по условию задачи входные точки являются попарно различными и не лежат на одной прямой, они гарантированно образуют невырожденный треугольник. Поэтому для решения задачи достаточно вычислить по известным координатам вершин на плоскости длины всех сторон треугольника, а затем проверить, выполняется ли для этих длин сторон теорема Пифагора: квадрат наибольшей по длине стороны должен быть равен сумме квадратов двух других сторон.

При решении этой задачи могут возникнуть две сложности:

1) Вещественное сравнение.
   Поскольку длины сторон a, b, c в общем случае будут вещественны, равенство $a^2 = b^2+c^2$ для предварительно вычисленных вещественных значений может не выполняться точно. Причина в том, что вещественные числа в памяти компьютера хранятся с некоторым приближением (величина его зависит от конкретного языка программирования и используемого типа) – и из-за округлений в процессе вычисления значения a^2 и b^2+c^2 будут немного, но различаться. То есть, выражение

   if ($a^2 = b^2+c^2$) then …

   окажется ложным, хотя при вычислении с абсолютной точностью оно должно было бы быть истинным.

   Возможных решений данной проблемы два. Первое - при сравнении вещественных чисел использовать сравнение с заданной точностью. Выражение

   if ($a^2 = b^2+c^2$) then …

   нужно в этом случае переписать в виде:

   if abs($a^2 – b^2 – c^2$) < ε then …

   где ε – предварительно заданная константа, равная, например, 10^{-9}.

   Сравнение с заданной точностью зачастую является допустимым решением, но иногда также может приводить к ошибкам (когда два близких, но отличающихся вещественных числа будут при таком сравнении приняты равными). Поэтому более корректным является второй способ - решение задачи в целых числах. В данном случае для проверки теоремы Пифагора нам нужны не сами длины сторон треугольника, а их квадраты: a^2, b^2, c^2 – которые при целочисленных координатах вершин всегда являются целыми числами. Поэтому достаточно вычислить, используя целочисленные типы данных, квадраты длин сторон треугольника, проверить для них выполнение теоремы Пифагора, и, если она выполняется, вычислить и вывести в ответ единственное вещественное число саму длину гипотенузы.

2) Вывод вещественного числа с заданной точностью.
   В каждом языке программирования существуют свои инструменты для вывода вещественных чисел в различных представлениях.

Задача E. «Салочки»

Согласно введенной в условии задачи метрике, расстояние между двумя регистрами A и B равно количеству различающихся бит в двоичных представлениях номеров этих регистров. Рассмотрим пример из условия (A=153, B=38017). Для некоторого регистра C суммарное расстояние AC+CB будет равно AC тогда и только тогда, когда все значения всех бит, совпадающих для A и B, те же самые и в двоичном представлении C:

```
153₁₀=    0000 0000 1001 1001₂
C =       X00X 0X00 100X X001
38017₁₀= 1001 0100 1000 0001₂
```

где на месте X может быть любое значение (0 или 1).
Если же хотя бы один из бит, совпадающих в A и B, для C отличается, то расстояние AC+CB будет всегда больше AC:

```
153₁₀=    0000 0000 1001 1001₂
C′ =      X00X 0X00 110X X001
38017₁₀= 1001 0100 1000 0001₂
```

Для решения задачи достаточно проверить для всех чисел во входных данных, не совпадающих с A и B, выполняется ли это условие. Для проверки условия можно явно строить двоичное представление чисел, используя операции деления нацело и остатка от деления (в данном случае на 2). Однако более просто данная задача решается с использованием битовой арифметики. В терминах битовой арифметики, регистр C удовлетворяет требованиям задачи, тогда и только тогда, когда верно следующее равенство:

(A xor B) = ((A xor C) or (C xor B))

Задача F. «Тандем»

Если T[i] – количество зубьев соответствующей шестеренки, ω_I, ω_{II}, ω_{III} – угловые скорости трех осей соответственно (согласно рисунку в условии), то связаны они будут следующими соотношениями:

$\omega_I * T[A] = \omega_{II} * T[B]$

$\omega_{II} * T[C] = \omega_{III} * T[D]$

Согласно условию задачи, должно выполняться:

1) $\omega_I = 2 * \omega_{III}$

2) $\omega_{II} < \omega_{III}$

Выражая эти соотношения через количество зубьев шестеренок, получаем условия:

1) $2 * T[A] * T[C] = T[B] * T[D]$

2) $T[C] > T[D]$

Как и в случае задачи «C», выполнение этих условий следует проверять в целых числах, избегая перехода к вещественным числам.

Малые количество неизвестных (4) и максимально возможное число шестеренок ($N \le 50$) позволяет искать решение задачи полным перебором всех возможных комбинаций шестеренок - то есть, 50*49*48*47= 5527200 комбинаций. Для еще большего ускорения работы алгоритма в перебор можно было внести отсечение, подбирая сначала пару шестеренок C и D и проверяя для неё выполнение условия T[C] > T[D], и лишь для подходящих пар продолжая поиск шестеренок A и B.