ВСЕРОССИЙСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ
по проверке и оцениванию решений задач регионального этапа Всероссийской олимпиады школьников по информатике в 2013/2014 учебном году

Утверждены Центральной предметно-методической комиссией по информатике 20 ноября 2013 г.

Москва 2013 г.

ОГЛАВЛЕНИЕ

Введение

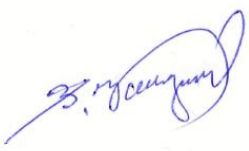
Для проведения регионального этапа Всероссийской олимпиады школьников по информатике тексты олимпиадных заданий, критерии и методики оценки выполненных олимпиадных заданий разрабатывает Центральная предметно-методическая комиссия по информатике (п.п. 16 и 41 Положения о Всероссийской олимпиаде школьников). С учетом этого Центральная предметно-методическая комиссия предоставляет организаторам и жюри регионального этапа 2013/2014 учебного года следующий комплект материалов:

- тексты олимпиадных задач;
- методику проверки решений задач, включая комплекты тестов для каждой задачи в электронном виде;
- проверяющие программы, позволяющие для каждой задачи определять правильность полученного решения в автоматическом режиме;
- описание системы оценивания решений задач;
- методические рекомендации по разбору предложенных олимпиадных задач.

Все вышеназванные материалы являются конфиденциальными и не подлежат преждевременному распространению до завершения регионального этапа по информатике во всех субъектах Российской Федерации.

Запрещается также размещать до начала или после завершения регионального этапа все или часть этих материалов на каких-либо региональных или личных сайтах без разрешения Департамента государственной политики в сфере общего образования Минобрнауки России.

Не допускается внесение каких-либо изменений или дополнений в представленные в распоряжение жюри регионального этапа материалы без согласования с Центральной предметно-методической комиссией по информатике. В случае возникновения со стороны регионального жюри каких-либо вопросов или замечаний по условиям задач или системы оценивания необходимо обращаться в Центральную предметно-методическую комиссию по информатике. Для оперативной связи с комиссией можно использовать адреса электронной почты: vkiryukhin@nmg.ru , poromov@gmail.com .

Председатель Центральной предметно-методической комиссии по информатике  В.М. Кирюхин

1. Комплект олимпиадных задач

Для проведения регионального этапа олимпиады по информатике Центральная предметно-методическая комиссия разработала комплект из восьми задач. Этот комплект задач является единым для всех участников регионального этапа, независимо от класса, в котором они обучаются. Обусловлено это тем фактом, что в большинстве случаев эти задачи являются многоуровневыми, но не с точки зрения формулировки условия, а с точки зрения возможных методов ее решения в зависимости от размерности задачи.

Все задачи пронумерованы от 1 до 8. Задачи с 1-й по 4-ю включительно предназначены для проведения первого тура, а задачи с 5-й по 8-ю включительно – для проведения второго тура. Номера задач соответствуют их сложности, например, задачи с номерами 1 и 5 являются, по мнению Центральной предметно-методической комиссии, простейшими и должны быть доступны практически всем участникам регионального этапа. В свою очередь, задачи с номерами 4 и 8 являются задачами повышенной сложности и ориентированы на сильнейших участников.

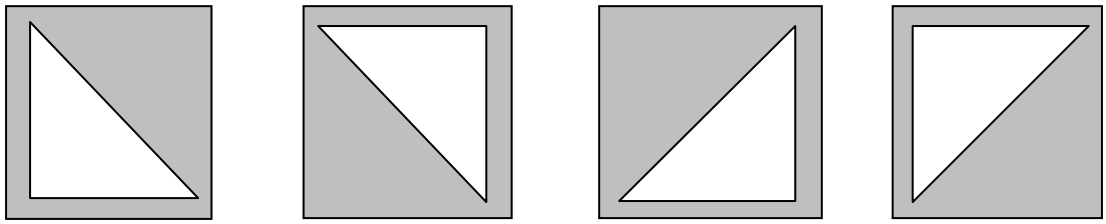
Непосредственно перед началом каждого тура тексты задач для этого тура должны быть растиражированы в количестве экземпляров, соответствующем числу участников олимпиады. На компакт-диске, который также предоставляется организаторам регионального этапа, в папке «Тексты задач для размножения» содержатся готовые для размножения условия задач для каждого тура. Участники должны получить доступ к текстам задач только в момент начала тура.

До окончания тура **категорически запрещается** распространять тексты задач за пределами мест размещения участников олимпиады. Учителя, тренеры, наставники и другие заинтересованные лица могут ознакомиться с содержанием олимпиадных задач тура в местах проведения регионального этапа только после начала тура во всех субъектах Российской Федерации, т.е. после 10.00 по московскому времени в день его проведения.

Задача 1. «POBEDA-2014»

Имя входного файла:	`pobeda.in`
Имя выходного файла:	`pobeda.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Как известно, современные видеокарты умеют формировать изображения с использованием только треугольников. Видеокарта POBEDA-2014 не отстает от современных тенденций. Известно, что она умеет отображать только прямоугольные



1 тип *2 тип* *3 тип* *4 тип*

равнобедренные треугольники четырех типов ориентации, представленные на рисунках ниже. Изменять ориентацию этих треугольников видеокарта не может.

Длина катета каждого из представленных выше треугольников равна одному сантиметру. За один такт видеокарта не может отобразить более чем a_i треугольников i-того типа.

Необходимо определить максимально возможную длину стороны квадрата, который может быть изображен видеокартой на экране монитора за один такт. При этом квадрат должен быть расположен так, чтобы его стороны были параллельны краям монитора.

Требуется написать программу, которая решает поставленную задачу.

Формат входного файла

Первая строка входного файла содержит разделенные пробелами четыре целых числа: a_1, a_2, a_3, a_4 ($0 \le a_1, a_2, a_3, a_4 \le 10^{18}$). Входные данные могут превышать максимальные значения для 32 битного типа данных.

Формат выходного файла

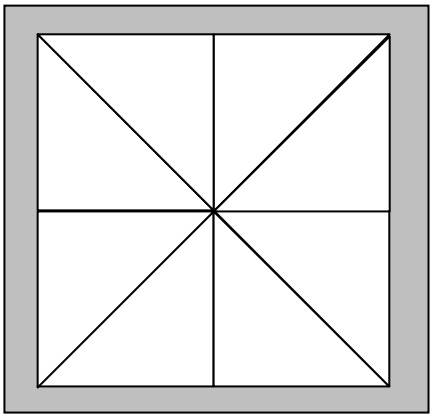
Выходной файл должен содержать одно число – максимально возможную длину стороны квадрата.

Примеры входных и выходных файлов

pobeda.in	pobeda.out
2 2 2 2	2
10 10 0 0	3

Пояснения к примерам

Далее приведен рисунок для первого примера.



Система оценивания

Частичные правильные решения для тестов, в которых $a_1, a_2, a_3, a_4 \le 100\,000$, будут оцениваться из 50 баллов.

Задача 2. «Список школ»

Имя входного файла: schools.in
Имя выходного файла: schools.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

При регистрации на портале интернет-олимпиады все участники заполняют регистрационную форму, где они указывают название школы, в которой они учатся. Разные участники могут по-разному писать название школы, например, «Физико-математическая школа №18», «ФМШ №18».

Организаторам олимпиады предоставлена информация о названиях школ, которые написали регистрируемые участники олимпиады. Точно известно, что цифры в названии школы встречаются только в номере школы, а число в записи названия школы встречается ровно один раз и оно однозначно определяет номер школы. Номер школы является положительным целым числом и не может начинаться с нуля.

Требуется написать программу для сайта интернет-олимпиады, которая поможет организаторам олимпиады получить следующую информацию: количество школ и номера школ, из которых зарегистрировалось не более пяти участников.

Формат входного файла

Первая строка входного файла содержит одно целое число n ($1 \le n \le 1000$) – количество названий школ, указанных всеми участниками при регистрации.

Последующие n строк содержат названия школ, указанные участниками. Название школы содержит только заглавные и строчные буквы латинского алфавита, цифры и пробелы, длина названия не превышает 100 символов.

Формат выходного файла

Первая строка выходного файла должна содержать одно число m – количество школ, от которых на олимпиаду зарегистрировалось от одного до пяти участников. Последующие m строк должны содержать только номера таких школ, при этом номера должны располагаться по одному в строке в произвольном порядке.

Примеры входных и выходных файлов

schools.in	schools.out
9 Physics and Mathematics School 18 9ya shkola imeni Pushkina Lyceum 9 PaMS 18 Gymnasium 42 School 9 Shkola nomer 9 High school 9 School N 9	2 42 18

Пояснения к примерам

В приведенном примере для участия в интернет-олимпиаде зарегистрировались: два ученика из школы с номером 18, один ученик из школы с номером 42 и шесть учеников из школы с номером 9. Таким образом, от 1 до 5 участников зарегистрировано от школ с номерами 18 и 42.

Система оценивания

Частичные правильные решения для тестов, в которых все номера школ являются однозначными числами, будут оцениваться из 30 баллов.

Частичные правильные решения для тестов, в которых номера школ – это числа строго меньше 1000, будут оцениваться из 50 баллов.

Частичные правильные решения для тестов, в которых номера школ – это числа строго меньше 10^9, будут оцениваться из 80 баллов.

Несмотря на выделение отдельных групп тестов, на окончательную проверку будут приниматься только решения, правильно работающие для теста из условия задачи.

Задача 3. «Межрегиональная олимпиада»

Имя входного файла:	`olympiad.in`
Имя выходного файла:	`olympiad.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

На межрегиональной олимпиаде по программированию роботов соревнования проводятся в один тур и в необычном формате. Задачи участникам раздаются последовательно, а не все в самом начале тура, и каждая i-я задача ($1 \le i \le n$) становится доступной участникам в свой момент времени s_i. При поступлении очередной задачи каждый участник должен сразу определить, будет он ее решать или нет. В случае, если он выбирает для решения эту задачу, то у него есть t_i минут на то, чтобы сдать ее решение на проверку, причем в течение этого времени он не может переключиться на решение другой задачи. Если же участник отказывается от решения этой задачи, то в будущем он не может к ней вернуться. В тот момент, когда закончилось время, отведенное на задачу, которую решает участник, он может начать решать другую задачу, ставшую доступной в этот же момент, если такая задача есть, или ждать появления другой задачи. При этом за правильное решение i-й задачи участник получает c_i баллов.

Артур, представляющий на межрегиональной олимпиаде один из региональных центров искусственного интеллекта, понимает, что важную роль на такой олимпиаде играет не только умение решать задачи, но и правильный стратегический расчет того, какие задачи надо решать, а какие пропустить. Ему, как и всем участникам, до начала тура известно, в какой момент времени каждая задача станет доступной, сколько времени будет отведено на ее решение и сколько баллов можно получить за ее решение. Артур является талантливым школьником и поэтому сможет успешно решить за отведенное время и сдать на проверку любую задачу, которую он выберет для решения на олимпиаде.

Требуется написать программу, которая определяет, какое максимальное количество баллов Артур сможет получить при оптимальном выборе задач, которые он будет решать, а также количество и перечень таких задач.

Формат входного файла

Первая строка входного файла содержит одно целое число n ($1 \le n \le 100\,000$) - количество задач на олимпиаде.

Последующие n строк содержат описания задач, по три числа на каждой строке: s_i - момент появления i-й задачи в минутах, t_i - время, отведенное на ее решение в минутах, и c_i - сколько баллов получит участник за решение этой задачи ($1 \le s_i, t_i, c_i \le 10^9$).

Формат выходного файла

Первая строка выходного файл должна содержать одно число – максимальное количество баллов, которое сможет получить Артур на олимпиаде.

Вторая строка должна содержать одно целое число m - количество задач, которые надо решить при оптимальном выборе.

Третья строка должна содержать m разделенных пробелом целых чисел - номера этих задач в порядке их решения. Задачи пронумерованы, начиная с единицы, в порядке их описания во входном файле.

Если оптимальных ответов несколько, необходимо вывести любой из них.

Примеры входных и выходных файлов

olympiad.in	olympiad.out
2 1 1 1 2 2 2	3 2 1 2
3 1 2 1 3 2 1 2 4 3	3 1 3

Пояснения к примерам

В первом примере Артур успевает решить все задачи и получить три балла.

Во втором примере Артуру выгоднее решать последнюю задачу и получить за нее три балла, чем решать только первые две и получить два балла.

Система оценивания

Частичные правильные решения для тестов, в которых все c_i одинаковы и $n \le 1000$, оцениваются из 30 баллов.

Частичные правильные решения для тестов, в которых все c_i одинаковы, оцениваются из 50 баллов.

Частичные правильные решения для тестов, в которых $n \le 1000$, оцениваются из 50 баллов.

Задача 4. «Дом Мэра»

Имя входного файла:	`majorhouse.in`
Имя выходного файла:	`majorhouse.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

При планировании нового района города М было решено, что дороги в новом районе будут образовывать прямоугольную сетку, то есть все улицы будут одного из двух типов – направленные с юга на север или направленные с востока на запад. При этом параллельные улицы будут проходить через каждый километр, и каждый квартал будет иметь размер ровно один километр на один километр. Таким образом, все дороги образуют равномерную клетчатую сетку. По каждой дороге разрешен проезд в любом из двух направлений.

Через некоторое время после постройки дорог оказалось, что такая планировка не всегда удобна, поскольку для постройки больших заводов или других сооружений и организации парков недостаточно одного квартала. Мэрия города М решила отдать каждому большому проекту по прямоугольному блоку из нескольких соседних кварталов. К сожалению, после реализации проекта все дороги внутри такого блока будут закрыты для проезда, но по границе блоков проезд все еще будет возможен. Касание двух блоков не закрывает проезд между ними.

Когда Мэру города М принесли на согласование план распределения территорий для больших проектов, ему стало интересно, насколько сложным будет маршрут от мэрии до его будущего дома. Мэрия находится в центре нового района, на пересечении нулевой улицы, направленной с юга на север, и нулевой улицы, направленной с востока на запад. С итоговым расположением дома Мэр еще не определился и на выбор у него есть k вариантов. Каждый из вариантов находится на пересечении x_i-ой улицы, направленной с юга на север (положительный x означает, что улица находится восточнее мэрии, отрицательный – западнее) и y_i-ой улицы, направленной с востока на запад (положительный y означает, что улица находится севернее мэрии, отрицательный - южнее).

Мэр считает, что маршрут до дома является сложным, если ему на этом маршруте придется совершить более двух поворотов направо или налево. Машина Мэра не может совершить более одного поворота на перекрестке, например, чтобы развернуться. Длина маршрута не имеет значения, и к дому можно подъезжать с любой стороны. Машина Мэра

всегда стоит у мэрии в северном направлении, может повернуть сразу направо или налево, но не может развернуться.

Требуется написать программу, которая по информации о закрытых для проезда блоках кварталов и возможным расположениям дома Мэра, для каждого возможного расположения дома Мэра найдет несложный маршрут от мэрии до дома, определит кратчайший из них, или сообщит, что такого маршрута не существует. Количество поворотов минимизировать не требуется.

Формат входного файла

Первая строка входного файла содержит два целых числа n и k ($0 \le n \le 100\,000$, $1 \le k \le 10$) – количество блоков кварталов, которые по плану будут отданы большим проектам и количество вариантов расположения дома Мэра, соответственно.

Последующие n строк содержат по описанию блоков кварталов - четыре целых числа u_1, v_1, u_2, v_2 ($-10^9 \le u_1 < u_2 \le 10^9$, $-10^9 \le v_1 < v_2 \le 10^9$) — номера улиц, на пересечении которых расположены противоположные углы блока кварталов, отданных под застройку и закрытых для проезда.

Последующие последние k строк содержат по два целых числа x_i и y_i ($|x_i| \le 10^9$, $|y_i| \le 10^9$, $x_i \ne 0$ или $y_i \ne 0$) — возможные расположения дома Мэра.

Мэрия и никакое из возможных расположений дома Мэра не находятся внутри блоков кварталов, отданных под застройку, но блоки кварталов, отданные под застройку, могут пересекаться.

Формат выходного файла

В выходной файл для каждого из возможных расположений дома Мэра в порядке появления во входном файле необходимо вывести сообщение, существуют ли несложный маршрут от мэрии до дома Мэра и, если существует, то где надо сделать повороты.

Если не существует несложный маршрут, то сообщение должно содержать слово NO на одной строке. Иначе, в первой строке должно содержаться слово YES, во второй строке – одно число t ($0 \le t \le 2$) - количество поворотов, и в последующих t строках – описания поворотов в порядке их совершения: в каждой строке по три числа x, y и d - номера улиц, на которых расположен перекресток, где необходимо повернуть, и направление поворота, при этом $d = -1$ означает поворот налево и $d = 1$ – поворот направо.

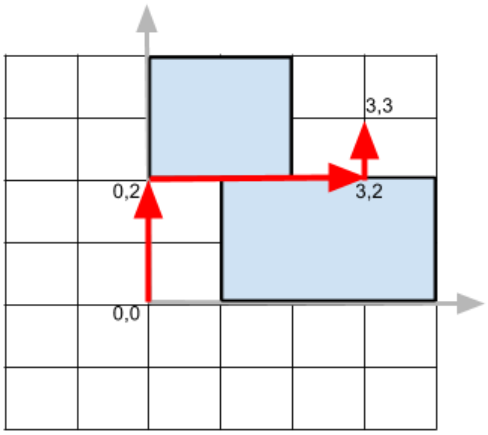
Координаты перекрестков, где необходимо совершить повороты, не должны превышать 10^9. Если кратчайших несложных путей несколько, необходимо вывести любой из них.

Примеры входных и выходных файлов

majorhouse.in	majorhouse.out
1 2 -2 1 9 2 2 0 3 3	YES 1 0 0 1 NO
2 1 0 2 2 4 1 0 4 2 3 3	YES 2 0 2 1 3 2 -1
0 2 0 -1 0 1	NO YES 0

Пояснения к примерам

Далее приведен рисунок для второго примера.



Система оценивания

Частичные правильные решения для тестов, в которых все координаты (x, y, u и v) по модулю не превышают 100, и $n \le 50$, будут оцениваться из 30 баллов.

Частичные правильные решения для тестов, в которых $n \le 50$, будут оцениваться из 60 баллов.

Задача 5 «Светофоры»

Имя входного файла: `lights.in`
Имя выходного файла: `lights.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

По территории компьютерного лагеря проложен маршрут для поездок на электрокарах. Поскольку на электрокаре можно добраться до ИКТ-центра, то школьник Пахом решил воспользоваться им. Следуя по маршруту, электрокар проехал с постоянной скоростью один за другим два светофора с зеленым светом. Пахому известно, что оба светофора находятся на расстоянии x метров друг от друга и переключаются абсолютно

синхронно: зеленый свет горит a минут, потом включается красный свет и горит в течение b минут, после чего светофор переключается опять на зеленый свет и он горит также в течение a минут, и так далее. Переключений на желтый свет у светофоров нет. Скорость движения электрокара по маршруту не превышает 1000 м/мин. Электрокар может проехать на светофоре в тот момент, когда светофор переключается с одного света на другой.

Приехав в ИКТ-центр, Пахом заинтересовался, с какой максимальной постоянной скоростью он мог ехать на электрокаре между двумя светофорами.

Требуется написать программу, которая позволит Пахому выяснить это.

Формат входного файла

Первая строка входного файла содержит три целых числа: a, b и x ($1 \le a \le 100$, $1 \le b \le 100$, $1 \le x \le 100\,000$).

Формат выходного файла

Выходной файл должен содержать одно число – максимальную возможную скорость электрокара между двумя светофорами. Ответ должен отличаться от правильного не более чем на 10^{-9}.

Примеры входных и выходных файлов

lights.in	lights.out
3 5 4000	800
5 10 21010	840.4

Система оценивания

Правильные решения для тестов, в которых ответ является целочисленным, будут оцениваться из 50 баллов.

Несмотря на выделение отдельной группы тестов для целочисленных ответов, на окончательную проверку будут приниматься только решения, правильно работающие для всех тестов из условия задачи.

Задача 6. «Кондиционеры»

Имя входного файла: `cond.in`
Имя выходного файла: `cond.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

При реализации проекта «Умная школа» было решено в каждый учебный класс выбранной для этого школы установить по кондиционеру нового поколения для

автоматического охлаждения и вентиляции воздуха. По проекту в каждом классе должен быть установлен только один кондиционер и мощность кондиционера должна быть достаточной для размеров класса. Чем больше класс, тем мощнее должен быть кондиционер.

Все классы школы пронумерованы последовательно от 1 до n. Известно, что для каждого класса с номером i, требуется ровно один кондиционер, мощность которого больше или равна a_i ватт.

Администрации школы предоставили список из m различных моделей кондиционеров, которые можно закупить. Для каждой модели кондиционера известна его мощность и стоимость. Требуется написать программу, которая определит, за какую минимальную суммарную стоимость кондиционеров можно оснастить все классы школы.

Формат входного файла

Первая строка входного файла содержит одно целое число n ($1 \le n \le 50\,000$) - количество классов в школе.

Вторая строка содержит n целых чисел a_i ($1 \le a_i \le 1000$) - минимальная мощность кондиционера в ваттах, который можно установить в классе с номером i.

Третья строка содержит одно целое число m ($1 \le m \le 50\,000$) - количество предложенных моделей кондиционеров.

Далее, в каждой из m строк содержится пара целых чисел b_j и c_j ($1 \le b_j \le 1000$, $1 \le c_j \le 1000$) - мощность в ваттах j-й модели кондиционера и его цена в рублях соответственно.

Формат выходного файла

Выходной файл должен содержать одно число - минимальную суммарную стоимость кондиционеров в рублях. Гарантируется, что хотя бы один корректный выбор кондиционеров существует, и во всех классах можно установить подходящий кондиционер.

Примеры входных и выходных файлов

cond.in	cond.out
1 800 1 800 1000	1000
3 1 2 3 4 1 10 1 5 10 7 2 3	13

Пояснения к примерам

В первом примере нужно купить один единственно возможный кондиционер за 1000 рублей.

Во втором примере оптимально будет установить в первом и втором классах кондиционеры четвертого типа, а в третьем классе – кондиционер третьего типа. Суммарная стоимость этих кондиционеров будет составлять 13 рублей (3 + 3 + 7).

Система оценивания

Частичные решения для $n, m \leq 1000$ будут оцениваться из 50 баллов.

Задача 7. «Конфеты»

Имя входного файла: `sweets.in`
Имя выходного файла: `sweets.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

Кондитерская фабрика города П, в котором живет Петя делает очень вкусные конфеты. Как-то раз, Петя собрался в гости к своему другу Васе, который живет в городе М. От города П до города М Петя решил доехать на поезде и взять с собой в подарок как можно больше коробок вкусных конфет.

Каждая коробка конфет имеет размеры $a \times b \times c$ сантиметров, где a – длина, b – ширина и c – высота коробки. Для перевозки конфет Петя хочет использовать один большой ящик в форме прямоугольного параллелепипеда. В ящик должны быть уложены все коробки конфет. Для того чтобы не повредить их, все коробки в ящике должны сохранять исходную ориентацию и располагаться в одном направлении. Петя может использовать ящик любого размера, но по правилам железнодорожных перевозок размер ящика по сумме трех измерений не может превышать N сантиметров.

Требуется написать программу, которая по заданным числам N, a, b и c определяет такой размер ящика, который должен использовать Петя, чтобы в него поместилось максимальное количество коробок конфет.

Формат входного файла

Первая строка входного файла содержит разделенные пробелами четыре целых числа: N, a, b, c ($1 \le N, a, b, c \le 10^9$).

Формат выходного файла

Выходной файл должен содержать три числа – длину, ширину и высоту ящика, который должен выбрать Петя и в который поместится максимальное количество коробок конфет. Если подходящих ответов несколько, необходимо вывести любой.

Примеры входных и выходных файлов

sweets.in	sweets.out
10 1 2 3	3 4 3
14 8 3 2	9 3 2

Пояснения к примерам

В первом примере выгоднее всего взять ящик размером $3 \times 4 \times 3$ сантиметров, в который поместится три коробки конфет в длину, две коробки конфет в ширину и одна коробка конфет в высоту.

Во втором примере для того, чтобы разместить хотя бы две коробки конфет, нужен ящик размером хотя бы $8 \times 3 \times 4$, у которого сумма измерений равна 15. В подходящий ящик поместится максимум одна коробка конфет. Подходящим также является ящик размером $9 \times 3 \times 2$, хотя он и не является минимальным.

Система оценивания

Частичные правильные решения для тестов, в которых $N \le 1000$, будут оцениваться из 30 баллов.

Частичные правильные решения для тестов, в которых $N \le 100\,000$, будут оцениваться из 60 баллов.

Задача 8. «Волонтеры»

Имя входного файла:	`volunteers.in`
Имя выходного файла:	`volunteers.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В проведении межпланетной олимпиады по информатике принимает участие большое количество андроидов. Андроиды, придумывающие задачи, пишущие условия и разрабатывающие наборы тестов, являются членами научного комитета. Те андроиды, которые обеспечивают работоспособность компьютеров и установленного на них программного обеспечения, являются членами технического комитета. Есть третья большая группа андроидов, участвующих в организации и проведении многих других мероприятий, – это волонтеры.

У каждого волонтера есть свой непосредственный начальник в научном комитете и свой непосредственный начальник в техническом комитете. У каждого члена научного комитета, кроме его председателя, есть свой непосредственный начальник – другой член научного комитета. У каждого члена технического комитета, кроме его председателя, также есть свой непосредственный начальник, являющийся членом технического комитета.

В самом начале олимпиады происходит процесс, определяющий начальников для всех волонтеров, членов научного комитета и членов технического комитета. Происходит он следующим образом.

Все n волонтеров получают уникальные номера от одного до n, после чего выстраиваются друг за другом в порядке возрастания своих номеров. После этого m членов научного комитета, также имеющие уникальные номера от одного до m, по очереди, в порядке возрастания их номеров, выбирают себе подчиненных следующим образом. Вначале, первый член научного комитета выбирает себе в подчинение некоторое количество волонтеров, стоящих в данный момент в строю подряд друг за другом. После этого те, кого он выбрал, выходят из строя, а он сам встает в строй на то место, где стояли его новоиспеченные подчиненные, заменяя собой сразу несколько андроидов в строю. Затем второй член научного комитета выбирает себе подчиненных из тех, кто теперь стоит в строю и заменяет их собой аналогично первому. Так этот процесс продолжается, и каждый последующий член научного комитета может выбрать себе в подчинение находящихся в строю волонтеров и членов научного комитета. Последним выбирает себе

подчиненных председатель научного комитета, и он должен выбрать всех андроидов, кто на тот момент будет находиться в строю.

После того, как в строю останется только председатель научного комитета, аналогичный процесс повторяется с самого начала, но уже с членами технического комитета. Важно то, что номера волонтеров при этом сохраняются и волонтеры выстраиваются в том же порядке, что и при выборе подчиненных членами научного комитета.

Считается, что член научного комитета может конфликтовать с членом технического комитета, если один и тот же волонтер подчиняется, возможно, не напрямую, и члену научного комитета, и члену технического комитета. Из этого определения, в частности, следует, что председатель научного комитета всегда может конфликтовать с председателем технического комитета. Несложно понять, что от общего количества таких возможных конфликтов между андроидами зависит то, насколько качественно будет проведена олимпиада.

Требуется написать программу, которая для каждого члена научного комитета подсчитывает количество членов технического комитета, с которыми он может конфликтовать, и определяет суммарное количество таких конфликтов.

Формат входного файла

Первая строка входного файла содержит числа n, m и k ($1 \le n, m, k \le 100\,000$) – количество волонтеров, членов научного комитета и членов технического комитета соответственно.

Последующие n строк содержат по два числа a_i и b_i ($1 \le a_i \le m$, $1 \le b_i \le k$) – номер начальника соответствующего волонтера среди членов научного комитета и номер начальника соответствующего волонтера среди членов технического комитета.

Следующая строка содержит (m–1) чисел c_i ($i = 0..(m–1)$, $i < c_i \le m$) – номер начальника i-го члена научного комитета. Председатель научного комитета имеет номер m.

Следующая строка содержит (k-1) чисел d_i ($i = 0..(k–1)$, $i < d_i \le k$) – номер начальника i-го члена технического комитета. Председатель технического комитета имеет номер k.

Гарантируется, что все входные данные корректны, то есть они получены в результате процесса назначения подчиненных, описанного выше.

Формат выходного файла

Первая строка выходного файла должна содержать одно число – суммарное количество искомых конфликтов.

Примеры входных и выходных файлов

volunteers.in	volunteers.out
4 3 4 1 2 1 1 2 1 2 4 3 3 3 3 4	11

Пояснения к примерам

В представленном примере только второй член научного комитета и второй член технического комитета не могут конфликтовать друг с другом.

Система оценивания

Частичные правильные решения для тестов, в которых количества волонтеров, членов научного комитета и членов технического комитета не превышают 100, будут оцениваться из 30 баллов.

Частичные правильные решения для тестов, в которых количества волонтеров, членов научного комитета и членов технического комитета не превышают 5000, будут оцениваться из 60 баллов.

2. Методика проверки решений задач

Для проверки решений участников Центральная предметно-методическая комиссия по информатике подготовила для регионального жюри следующие материалы:

- общие методические рекомендации по проверке решений участников;
- комплекты тестов в электронном виде, содержащие для каждой задачи файлы входных данных и соответствующие им файлы выходных данных (для некоторых задач имеются специально разработанные программы – генераторы тестов);
- проверяющие программы, позволяющие для каждой задачи определять правильность полученного решения в автоматическом режиме и программы визуализаторы тестов;
- эталонное решение по каждой задаче для отладки системы автоматической проверки;

Все перечисленные выше материалы для каждой задачи представлены на компакт-диске, который поступает в распоряжение организаторов регионального этапа олимпиады до начала регионального этапа. Материалы для непосредственной проверки решений каждой задачи размещены на этом компакт-диске в отдельной папке с соответствующим именем. Эта папка содержит:

1) Папку «preliminary», содержащую тесты из примеров в условии задачи, предназначенные для предварительной проверки решений участника во время тура. Каждый тест из примера содержится в отдельном файле, входные файлы называются «01», «02», и т.д. Файлы с правильными ответами называются «01.a», «02.a», и т.д. Тесты пронумерованы в том же порядке, в котором они следуют в условии задачи.

2) Папку «tests», содержащую тесты для окончательной проверки и оценивания решений участников, и правильные ответы. Каждый тест содержится в отдельном файле. Входные файлы называются «01», «02» и т.д. Файлы с правильными ответами называются «01.a», «02.a» и т.д.

3) Папку «src», содержащую программы-генераторы, использовавшиеся для создания тестов. Программы-генераторы приведены исключительно для ознакомления жюри регионального этапа с методами получения тестов. Их запуск и использование для перегенерации тестов не требуется, так как комплект материалов уже содержит сгенерированные тесты и ответы к ним. Программы генераторы содержатся в файлах, написанных на различных языках программирования, их компиляция и метод использования описаны в командном файле «buildTests.cmd».

4) Файлы, содержащие проверяющие программы для проверки решений участников с использованием специализированных проверяющих программных систем. Проверяющая программа может содержаться либо в файле «check.dpr», либо в файле «check.cpp». Для компиляции файла «check.dpr» следует использовать компилятор Borland Delphi. Кроме того, при ее исполнении используется библиотека для проверяющих программ testlib, которая содержится в папке «lib» в файле «testlib.pas». Для компиляции файла «check.cpp» можно использовать GNU C++ или Visual C++. Используемая в этом случае библиотека «testlib.h» также содержится в папке «lib».

5) Папку «main», содержащую основное решение жюри. Оно должно использоваться для установки ограничений по времени (см. раздел 2.1, пункт 3).

6) Примеры правильных и некорректных решений. Каждое решение находится в отдельном файле. Этот файл имеет имя, построенное по маске «problem_ab.ext», где «problem» – идентификатор задачи, обычно он совпадает с названием каталога, в котором находятся материалы задачи, «ab» – инициалы автора решения, «ext» – расширение файла, соответствующее языку программирования, на котором написано решение. Решения, которые являются некорректными или неоптимальными, также имеют суффикс, обычно указывающий на проблемы в этом решении, например «problem_ab_wrong.ext» (неправильное решение), «problem_ab_slow.ext» (медленное решение), «problem_ab_bug.ext» (решение, содержащее некоторую типичную ошибку).

Решения предоставляются только для ознакомления членов регионального жюри с возможной реализацией правильных и неправильных решений, а также для проверки работоспособности проверяющих систем. Их использование для генерации правильных ответов на тесты не требуется, так как соответствующие материалы уже содержат готовые правильные ответы на все тесты.

2.1. Общие методические рекомендации по проверке решений задач

При проверке решений участников необходимо учитывать следующее.

1. Всем участникам регионального этапа Всероссийской олимпиады школьников, независимо от классов обучения, на каждом туре предлагается один и тот же комплект из четырех задач.

2. Текст каждой задачи содержит описание задачи, максимальное время работы программы на отдельном тесте, размер доступной программе памяти в процессе ее

исполнения, форматы входных и выходных данных и примеры входных и выходных данных, варианты оценивания частичных решений.

3. Заданные в условии каждой задачи ограничения на максимальное время работы программы ориентированы на использование при проверке решений участников компьютеров с процессором Core 2 Duo 2.4 ГГц. Именно такого типа компьютеры необходимо использовать для проверки решений участников, причем все решения должны проверяться на одинаковых компьютерах.

Если в распоряжении жюри окажутся компьютеры с другим быстродействием, то рекомендуется установить ограничение по времени в каждой задаче в два раза больше, чем максимальное время работы решения жюри на одном тесте, но не менее 1 секунды. Например, если есть компьютер с процессором Pentium 3 866 МГц и решение жюри работает максимум 2 секунды, то необходимо установить ограничение, равное 4 секундам. В свою очередь, если есть компьютер с процессором Core i7 3 ГГц и время работы решения жюри равно 0,2 секунды, то следует задать ограничение, равное 1 секунде.

4. Результатом решения всех предложенных задач является исходный текст программы на одном из языков программирования, принадлежащем основной или дополнительной группе (см. «Требования к организации и проведению регионального этапа Всероссийской олимпиады школьников по информатике в 2013/2014 учебном году»). Центральная предметно-методическая комиссия по информатике предоставляет региональному жюри материалы для проверки решений только для языков и сред программирования основной группы. При использовании организаторами регионального этапа языков и сред программирования дополнительной группы не гарантируется возможность получения полного решения олимпиадных задач регионального этапа.

5. Проверка решений участников осуществляется путем исполнения программы с входными данными, соответствующими каждому тесту из предложенного Центральной предметно-методической комиссией по информатике комплекта тестов с последующим анализом получаемых в результате этого выходных файлов.

6. Поскольку участники олимпиады должны сдавать на проверку решения в виде исходного текста программы на одном из допустимых языков программирования, то проверка решений каждого участника должна осуществляться в следующей последовательности:

- компиляция исходного текста программы;

- последовательное исполнение полученного exe-файла для файлов с входными данными, соответствующих тестам из набора тестов для данной задачи,
- сравнение результатов исполнения программы на каждом тесте с правильным ответом.

При компиляции исходного текста программы, которую участник сдал на проверку, необходимо учитывать следующее.

1) Жюри должно использовать вполне определенные командные строки для компиляции решений, о чем участников следует известить до начала тура. Эту информацию необходимо разместить в Памятке участнику, например, в следующем виде:

Компилятор	**Командная строка**
Free Pascal 2.6.0	fpc <исходный файл>
GNU C 4.6.1 (MinGW)	gcc -O2 -x c –Wl,--stack=67108864 <исходный файл>
Visual C++ 2010	cl /O2 /EHs /TP <исходный файл>
GNU C++ 4.6.1 (MinGW)	g++ -O2 -x c++ –Wl,--stack=67108864 <исходный файл>
Borland/Embarcadero Delphi 7.0	dcc32 -cc <исходный файл>

Жюри имеет право изменять команды компиляции решений в процессе проведения соревнований, но участники олимпиады должны быть обязательно проинформированы об этом перед началом тура.

2) Размер файла с исходным текстом программы не должен превышать **256** килобайт. Время компиляции программы не должно превышать **одной** минуты. В случае нарушения этих ограничений решение участника считается неправильным и никакие баллы за эту задачу участнику не начисляются. Информация об этих ограничениях также должна быть размещена в Памятке участнику.

3) При исполнении программы на каждом тесте, в первую очередь, жюри должно определить, нарушаются ли заданные в условии этой задачи ограничения на время работы программы на отдельном тесте и размер доступной программе памяти в процессе ее исполнения. В случае нарушения названных ограничений баллы за этот тест участнику не начисляются.

Если указанные выше ограничения в процессе исполнения программы с входными данными, соответствующими конкретному тесту, не нарушаются, то после завершения исполнения программы осуществляется проверка правильности полученного ответа. Эта

проверка осуществляется с использованием предоставляемых центральной предметно-методической комиссией по информатике проверяющих программ.

4) Все представленные на проверку решения участников сначала должны проходить предварительное тестирование на тесте или тестах из условия задачи. Если на этих тестах решение участника выдает правильный ответ, то тогда это решение принимается жюри на окончательную проверку на всех тестах из заданного набора тестов для этой задачи. В противном случае, решение участника считается неверным, и за него участнику не начисляются какие-либо баллы.

При проверке решений участников с использованием программной проверяющей системы процесс предварительной проверки осуществляется в течение тура по мере посылки решений на сервер соревнований. Результаты такой проверки сообщаются участнику сразу после ее завершения. В зависимости от возможностей проверяющей системы на окончательную проверку может приниматься либо последнее прошедшее предварительное тестирование решение одной и той же задачи, либо то, которое он должен указать. В любом случае, участник олимпиады должен быть проинформирован до начала тура, каким образом будет определяться решение, принимаемое системой для окончательной проверки. Эту информацию также следует разместить в Памятке участнику.

5) Окончательная проверка и оценивание решений участников может осуществляться либо после окончания тура, либо во время тура, если используемая жюри среда проведения соревнований позволяет это делать. Итоги такой проверки являются предварительными и доводятся индивидуально до сведения каждого участника только после завершения тура.

В заключение следует отметить, что результатом многократного исполнения программы участника с одними и теми же входными файлами должны быть одинаковые выходные файлы, вне зависимости от времени запуска программы и ее программного окружения. Региональное жюри вправе произвести неограниченное количество запусков программы участника и выбрать наихудший результат по каждому из тестов.

2.2. Характеристика тестов для каждой задачи

Комплект тестов для каждой задачи разрабатывался таким образом, чтобы региональное жюри могло в максимальной степени оценить все возможные типы алгоритмов, которые могут быть использованы в решениях участников, и

продифференцировать полученные участниками решения по степени их корректности и эффективности. В общем случае в комплекте тестов выделяются следующие группы тестов:

1) простые тесты;
2) тесты на частные случаи, позволяющие выявить особенности используемых алгоритмов;
3) общие тесты (достаточно случайные тесты, разные по размеру: от простых тестов до сложных);
4) тесты, проверяющие наличие эвристик в алгоритмах;
5) тесты максимальной размерности (тесты с использованием максимальных значений входных переменных, позволяющие оценить эффективность предложенных алгоритмов или их работоспособность при максимальной размерности задачи).

Количество тестов для каждой задачи различно и находится в диапазоне от 20 до 50. Файл с тестом называется «XY» и находится в каталоге tests. Тесты для каждой задачи пронумерованы от 1 до N, где N — количество тестов к задаче. Файл с правильным ответом на соответствующий тест называется «XY.a». Здесь XY — двузначный номер теста, дополненный при необходимости ведущим нулем. Например, файл с седьмым тестом называется «07», а файл с пятнадцатым тестом — «15», файл с правильным ответом на седьмой тест называется «07.a», файл с правильным ответом на пятнадцатый тест — «15.a».

Количество тестов, подготовленных для каждой задачи, представлено в таблице ниже.

Задача	**Количество тестов**
1. POBEDA-2014	20
2. Список школ	20
3. Межрегиональная олимпиада	50
4. Дом Мэра	50
5. Светофоры	50
6. Кондиционеры	20
7. Конфеты	50
8. Волонтеры	50

Как уже было сказано выше, наборы тестов для каждой задачи разработаны таким образом, чтобы жюри могло в максимальной степени оценить все возможные типы алгоритмов, которые могут быть использованы в решениях участников, и продифференцировать полученные участниками решения по степени их корректности и эффективности. Следует заметить, что правильное, но не эффективное решение задачи может набирать ориентировочно 30-70% баллов.

Ниже для каждой задачи описаны некоторые особенности оценивания возможных решений участников с использованием предложенных Центральной методической комиссией наборов тестов, которые будут полезны членам жюри и помогут лучше понять систему оценивания этих задач.

Задача 1 «POBEDA-2014»

Для проверки решений данной задачи выделены две группы тестов:

1) тесты, для которых все числа не превышают 100 000 (тесты 1 – 10);
2) тесты, в которых хотя бы одно число превышает 100 000 (тесты 11 – 20).

Вышеописанный набор тестов позволяет помимо полных решений оценивать частичные решения, не использующие функцию взятия корня из числа. Такие решения оцениваются из 50 баллов.

Задача 2 «Список школ»

Для проверки решений данной задачи выделены четыре группы тестов в зависимости от длины номеров школ:

1) тесты для однозначных номеров школ (тесты 1 – 6);
2) тесты для школ с номерами от 10 до 999 (тесты 7 – 10)
3) тесты для школ с номерами от 1000 до $(10^9–1)$ (тесты 11 – 16)
4) тесты для школ с номерами длиной более 9 символов (тесты 17 – 20)

Частичные решения, правильно работающие для первой группы тестов, оцениваются из 30 баллов.

Частичные решения, правильно работающие для двух первых групп тестов, могут набрать суммарно 50 баллов.

Частичные решения, правильно работающие для первых трех групп тестов, могут набрать суммарно 80 баллов.

Задача 3 «Межрегиональная олимпиада»

Для проверки решений данной задачи выделены четыре группы тестов, отличающиеся ограничениями на баллы за задачи, и количество задач:

1) тесты, в которых все задачи оцениваются одинаково и $1 < n \leq 1000$ (тесты 1 – 15);
2) тесты, в которых все задачи оцениваются одинаково и $1000 < n \leq 100000$ (тесты 16 – 25);
3) тесты, в которых задачи оцениваются разными баллами и $1 \leq n \leq 1000$ (тесты 26 – 35);
4) тесты, в которых задачи оцениваются разными баллами и $1000 < n \leq 100000$ (тесты 36 – 50).

Частичные решения, правильно работающие для первой группы тестов, оцениваются из 30 баллов.

Частичные решения, правильно работающие для двух первых групп тестов, могут набрать суммарно 50 баллов.

Частичные решения, правильно работающие для первой и третьей группы тестов, оцениваются из 50 баллов.

Задача 4 «Дом Мэра»

Для проверки решений данной задачи выделены три группы тестов в зависимости от ограничений на координаты и количество блоков кварталов:

1) тесты, в которых координаты по модулю не превышают 100, а количество блоков кварталов не превышает 50 (тесты 1-15);
2) тесты, в которых координаты по модулю превышают 100, а количество блоков кварталов не превышает 50 (тесты 16 – 30);
3) тесты, в которых количество блоков кварталов превышает 50 (тесты 31 – 50);

Вышеописанный набор тестов позволяет помимо полных решений оценивать следующие частичные решения данной задачи:

- решения, использующие поиск в ширину по всем перекресткам (оцениваются из 30 баллов);
- решения, использующие поиск в ширину по сжатым координатам (оцениваются из 60 баллов).

Задача 5 «Светофоры»

Для проверки решений данной задачи используется две группы тестов:

1) тесты, в которых ответ является целым числом (тесты 1 - 25);
2) тесты с вещественным (не целочисленным) ответом (тесты 26 – 50).

С помощью этих групп тестов, помимо полных решений, оцениваются также частичные решения, перебирающие возможные значения ответа и проверяющие, что он подходит. Такие решения оцениваются из 50 баллов.

Задача 6 «Кондиционеры»

Для проверки решений данной задачи выделены две группы тестов:

1) тесты, в которых количество моделей кондиционеров m и количество классов n не превышает 1000 (тесты 1 – 10);
2) тесты, в которых хотя бы одно из значений m и n больше 1000 (тесты 11 – 20).

Вышеописанный набор тестов позволяет помимо полных решений оценивать частичные решения, имеющие квадратичную асимптотическую сложность от количества моделей кондиционеров или классов. Такие решения оцениваются из 50 баллов.

Задача 7 «Конфеты»

Для проверки решений данной задачи выделены три группы тестов в зависимости от значения N - максимальной суммы измерений:

1) тесты, для которых $1 \le N \le 1000$ (тесты 1 – 15);
2) тесты, для которых $1000 < N \le 100\,000$ (тесты 16 – 30);
3) тесты, для которых $100\,000 \le N \le 10^9$ (тесты 31 – 50).

Частичные решения, правильно работающие для первой группы тестов, оцениваются из 30 баллов.

Частичные решения, правильно работающие для двух первых групп тестов, могут набрать суммарно 60 баллов.

Задача 8. «Волонтеры»

Для проверки решений данной задачи выделены три группы тестов в зависимости от количества волонтеров, членов научного и членов технического комитетов:

1) тесты, в которых количество андроидов в группах не превышает 100 (тесты 1 – 15);
2) тесты, в которых количество андроидов в группах принимает значения больше 100, но меньше 5000 (тесты 16 – 30);

3) тесты, в которых количество андроидов, хотя в одной группе, более 5000 (тесты 31 – 50);

Вышеописанный набор тестов позволяет помимо полных решений оценивать следующие частичные решения данной задачи:

- решения, имеющие асимптотическую сложность алгоритма $O(n^3)$ (оцениваются из 30 баллов), где n - количество андроидов в группах;
- решения, имеющие асимптотическую сложность алгоритма $O(n^2)$ (оцениваются из 60 баллов), где n - количество андроидов в группах.

3. Система оценивания решений участников

Система оценивания решений каждой задачи основана на следующих положениях:

1. Решение каждой задачи оценивается из 100 баллов, то есть, максимальное количество баллов, которое участник может получить за полное решение каждой задачи, составляет 100 баллов.

2. Общая оценка за решение отдельной задачи конкретным участником складывается из суммы баллов, начисленных ему по результатам исполнения всех тестов из набора тестов для этой задачи в процессе окончательной проверки всех решений после тура.

3. Итоговый результат каждого участника подсчитывается как сумма полученных этим участником баллов за решение каждой задачи на первом и втором турах. С учетом того факта, что на каждом из двух туров предлагается по четыре задачи, то максимально возможное количество баллов, которое может набрать участник по итогам регионального этапа, составляет 800 баллов.

Для предложенных Центральной предметно-методической комиссией по информатике задач регионального этапа оценка для каждого теста из комплекта тестов для каждой задачи является одинаковой. В таблице ниже эти оценки представлены.

Задача	Количество тестов	Оценка теста
1. POBEDA-2014	20	5 баллов
2. Список школ	20	5 баллов
3. Межрегиональная олимпиада	50	2 балла
4. Дом Мэра	50	2 балла
5. Светофоры	50	2 балла
6. Кондиционеры	20	5 баллов
7. Конфеты	50	2 балла
8. Волонтеры	50	2 балла

Окончательные результаты проверки решений всех участников фиксируются в трех итоговых таблицах – для обучающихся 9-х, 10-х и 11-х классов. Каждая таблица представляет собой ранжированный список участников соответствующего класса, расположенных по мере убывания набранных ими баллов. Участники с одинаковыми баллами располагаются в алфавитном порядке. На основании этих таблиц жюри принимает решение о победителях и призерах регионального этапа олимпиады по каждому классу с учетом квоты, установленной организатором регионального этапа.

4. Автоматизация процесса проверки и оценивания решений участников

Существуют различные способы проверки решений участников. Если по условию задачи ее решением должна быть программа, то самый простой способ, но в то же время самый трудоемкий, заключается в последовательном запуске проверяемой программы на каждом тесте из заданного комплекта тестов для этой задачи. Для этого способа вполне достаточно иметь для каждого теста файл с входными данными и файл с соответствующими выходными данными. Если учесть, что для каждой задачи эти файлы предоставляются региональному жюри центральной предметно-методической комиссией по информатике, то члены жюри вполне могут справиться с задачей проверки решений участников таким «ручным» способом при наличии достаточного количества членов жюри.

Конечно, описанный способ достаточно трудоемкий, но тот факт, что решения участников сначала проверяются на одном или двух тестах из условия задачи, и только в случае успешного прохождения этих тестов решение далее проверяется на всех тестах из заданного набора, в определенной степени уменьшает объем необходимой работы. Тем не менее, выходом из создавшегося положения является автоматизация процесса проверки решений участников.

В настоящее время во многих регионах уже успешно используются специализированные системы проведения различных соревнований по информатике. Они используются как при предварительной проверке решений участников во время тура, так и при окончательной проверке, которая осуществляется после завершения тура. Более того, уже де-факто сформировались вполне определенные к ним требования.

В частности, в процессе предварительной проверки решений участников, представленных в виде программ, такие системы должны последовательно выполнять следующие действия:

1) Скомпилировать программу участника, используя приведенную в Памятке участнику команду для соответствующего языка программирования. Если компиляция программы участника завершается неудачно, участнику сообщается результат «Ошибка компиляции». Возможно предоставление участнику вывода компилятора в стандартный поток вывода и стандартный поток ошибок. Если компиляция завершилась успешно, то далее программа проверяется на тестах из примера.

2) Осуществлять последовательную проверку программы участника на всех тестах из примера. Проверка на одном тесте осуществляется следующим образом. В пустой

каталог копируется исполняемый файл программы участника и тестовый входной файл. Тестовый файл должен иметь имя, указанное в условии задачи. Далее программа участника запускается, и проверяющая система отслеживает соблюдение программой существующих ограничений, связанных с запретом на создание каталогов и временных файлов при работе программы, осуществление чтения и записи векторов прерываний, а также любое использование сетевых средств и выполнение других действий, нарушающих работу самой проверяющей системы.

3) Обеспечивать контроль времени работы программы участника и объема используемой памяти. Если время работы программы превысило ограничение, указанное в условии задачи, выполнение программы участника прерывается и участнику отправляется сообщение «Превышено время работы». Если количество используемой памяти превысило ограничение, указанное в условии задачи, то выполнение программы участника также прерывается и участнику отправляется сообщение «Превышен максимальный объем используемой памяти».

4) Проверить, создала ли программа участника и самостоятельно обработала исключительную ситуацию (exception). Если программа участника создала и самостоятельно не обработала исключительную ситуацию (exception), выполнение программы участника прерывается и участнику отправляется сообщение «Ошибка времени исполнения».

5) Проверить, завершила ли программа участника работу с нулевым кодом возврата. Если программа участника завершила работу с ненулевым кодом возврата, участнику отправляется сообщение «Ошибка времени исполнения».

6) Проверить, создала ли программа участника в каталоге, в котором она была запущена, выходной файл с именем, указанным в условии задачи, если программа участника завершила работу за отведенный период времени, не превысила максимальный объем памяти и завершила работу с нулевым кодом возврата. Если файл с указанным именем не найден, участнику отправляется сообщение «Ошибка формата выходных данных». Если выходной файл создан, то осуществляется проверка его корректности. Для этого используется соответствующая проверяющая программа, которая предоставляется организаторам регионального этапа центральной предметно-методической комиссией по информатике.

7) Сообщить участнику о результатах проверки его программы. Если программа участника выдает правильный ответ на всех тестах из примера, то она может быть принята на окончательную проверку. В этом случае участнику отправляется сообщение

«Принято на проверку», а тестирующая система запоминает решение участника как последнее принятое решение по данной задаче. В противном случае участнику отправляется сообщение в соответствии с описанными выше правилами. При этом участнику помимо типа ошибки сообщается номер теста из примера, на котором произошла ошибка.

При окончательной проверке решений участников, представленных в виде программ, которая осуществляется после тура, программная система проведения олимпиад по информатике должна проверить на основных тестах последнее принятое на проверку решение участника по каждой задаче. Выполняемые системой функции в этом случае во многом повторяют вышеописанные. Однако при возникновении любой ошибки вместо отправки участнику сообщения, система просто помечает тест как не пройденный. Кроме того, по результатам окончательной проверки система начисляет участнику баллы за успешно пройденные тесты.

Для упрощения процесса использования систем, автоматизирующих проверку решений участников, Центральная предметно-методическая комиссия предоставляет в распоряжении регионального жюри специально разработанные для каждой задачи проверяющие программы.

Проверяющая программа для каждой задачи находится в папке с номером этой задачи либо в файле «check.dpr», либо в файле «check.cpp». Для компиляции файла «check.dpr» следует использовать компилятор Borland Delphi. Кроме того, при ее исполнении используется библиотека для проверяющих программ testlib, которая содержится в папке «lib» в файле «testlib.pas». Для компиляции файла «check.cpp» можно использовать GNU C++ или Visual C++. Используемая в этом случае библиотека «testlib.h» также содержится в папке «lib».

Запуск проверяющей программы осуществляется следующим образом. В пустой каталог необходимо скопировать проверяющую программу, входной файл, ответ для которого следует проверить (input), выходной файл, созданный программой участника (output) и файл с правильным ответом (answer). После этого проверяющая программа запускается с тремя параметрами командной строки — входной файл, выходной файл и файл с правильным ответом.

Программа завершает свою работу с одним из трех возможных кодов возврата: 0 — ответ участника является правильным; 1 — ответ участника удовлетворяет формату вывода, но является неправильным, в этом случае участнику отправляется сообщение

«Неверный ответ»; 2 — ответ участника не удовлетворяет формату вывода, в этом случае участнику отправляется сообщение «Ошибка формата выходных данных». Файл «result» после завершения программы будет содержать сформированный проверяющей программой комментарий о причинах, по которым был выдан соответствующий отклик. Этот комментарий не должен сообщаться участникам и служит только для справки членам регионального жюри.

Проверяющая программа check.exe используется как на этапе предварительной проверки, так и на этапе окончательной проверки и оценки решений участников. Отличие заключается в том, что при окончательной проверке при возникновении любой ошибки вместо формирования участнику сообщения, система просто помечает тест как не пройденный, и по результатам проверки участнику начисляются баллы за успешно пройденные тесты в соответствии с системой оценивания, разработанной центральной предметно-методической комиссии по информатике.

5. Краткие методические рекомендации по решению задач

При разработке задач для регионального этапа олимпиады по информатике Центральная предметно-методическая комиссия исходила из того, что все задачи должны быть оригинальными, быть разнообразными по тематике и не требовать для своего решения специальных знаний.

При определении уровня сложности задач разработчики исходили из того, что комплект задач должен содержать как задачи, доступные многим участникам регионального этапа, так и задачи, позволяющие проявить себя наиболее сильным участникам. В этой связи количество задач на каждом туре равно четырем, и как минимум две задачи из них такой сложности, что большинство школьников могут их решить, включая и школьников младших классов. Более того, все задачи являются многоуровневыми и предполагают наличие как полных, так и частичных решений, что также будет способствовать тому, что ни одна задача из предложенного комплекта не останется без внимания участников, а сильным участникам позволит продемонстрировать все свои лучшие качества.

Сказанное подтверждает общая характеристика всех задач, представленная в таблице ниже.

Задача	Тематика	Алгоритмическая сложность	Техническая сложность
1. POBEDA-2014	Математические основы информатики, геометрия	низкая	низкая
2. Список школ	Техника программирования	низкая	средняя
3. Межрегиональная олимпиада	Динамическое программирование	средняя	средняя
4. Дом Мэра	Сортировка, техника программирования	высокая	высокая
5. Светофоры	Математические основы информатики	низкая	низкая
6. Кондиционеры	Математические основы информатики, жадные алгоритмы	средняя	средняя
7. Конфеты	Теория чисел, перебор	высокая	средняя
8. Волонтеры	Теория графов, структуры данных	высокая	высокая

Далее приведены краткие методические указания по решению каждой задачи, которые помогут членам жюри выявить те или иные особенности существующих

решений, осмысленно подойти к проверке решений участников и подготовиться к разбору задач, который необходимо провести после завершения второго тура и получения результатов предварительной проверки решений всех участников.

Задача 1 «POBEDA-2014»

Основная идея решения данной задачи основана на следующем факте: для того, чтобы нарисовать единичный квадрат, требуется либо по одному треугольнику типа 1 и 2, либо по одному треугольнику типа 3 и 4. Таким образом, из a_1 треугольников типа 1 и a_2 треугольников типа 2 можно составить не более $\min\{a_1, a_2\}$ квадратов, а из a_3 треугольников типа 3 и a_4 треугольников типа 4 – $\min\{a_3, a_4\}$ квадратов.

Теперь, чтобы решить поставленную задачу, требуется определить, какой максимальный квадрат можно составить из $K = \min\{a_1, a_2\} + \min\{a_3, a_4\}$ единичных квадратов. Для этого можно либо перебирать все квадраты натуральных чисел (1, 4, 9, 16, …) до тех пор, пока не встретится число, большее K, либо извлечь из K квадратный корень и округлить результат до ближайшего целого вниз. Стоит учитывать малую точность операции взятия квадратного корня из большого числа в некоторых языках программирования, поэтому необходимо результат проверить обратным возведением в квадрат или использовать тип с расширенной точностью.

В первом случае сложность алгоритма будет равна $O(K^{1/2})$, а во втором – $O(1)$.

Задача 2 «Список школ»

При решении данной задачи необходимо сначала выделить номера школ из каждой записи с названием школы и сформировать массив, содержащий эти номера. Затем, используя полученный массив, определить, количество школ и номера школ, которые встречаются в нем не более пяти раз.

При выделении номера школы из ее названия *s* следует последовательно перемещаться по строке *s* слева направо, пока не встретится цифра. Это можно сделать с помощью следующего фрагмента программы:

```
i = 0
while s[i] not in '0123456789':
   i += 1
start = i
while i < len(s) and s[i] in '0123456789':
   i +=1
finish = i
```

После выполнения этого фрагмента программы подстрока строки *s* от символа с номером start до символа с номером finish (не включительно) и есть искомый номер школы. Следует выделить этот номер и сохранить его в массиве *sch*.

Поскольку номер школы может быть очень большим, то не следует переводить его в числовой тип данных, а сохранить как строку. Полученный массив номеров школ нужно отсортировать в лексикографическом (алфавитном) порядке. Теперь, чтобы получить искомый ответ, осталось подсчитать, сколько раз встречается в массиве каждая школа, и сформировать список школ, встречающихся не более 5 раз. Это можно сделать, например, с использованием следующего фрагмента программы:

```
count = 1
result = 0
answer = []
for i in range(1, len(sch)):
    if sch[i] == sch[i – 1]:
        count += 1
    else:
        if count <= 5:
            result += 1
            answer.append(sch[i-1])
        count = 0
if count <= 5: # не забываем последнюю школу
    result += 1
    answer.append(sch[-1])
```

Задача 3 «Межрегиональная олимпиада»

Разберем вначале частичное решение, основанное на предположении, что все задачи оцениваются одинаково. Представим каждую задачу отрезком на оси времени. Тогда задача сводится к классической: из заданного множества отрезков на прямой выбрать наибольшее количество отрезков, не имеющих общих точек.

Для решения этой задачи можно использовать следующий алгоритм. Вначале выберем из всех отрезков тот, который заканчивается левее всех. Из отрезков, которые начинаются правее него, опять выберем отрезок, заканчивающийся левее всех и т.д.

Не сложно определить, что реализация этого алгоритма имеет асимптотическую сложность $O(N^2)$. Более эффективное решение можно получить, отсортировав все начала и концы отрезков и пройдя слева направо по отсортированному массиву. Такое решение с учетом использования быстрой сортировки имеет уже асимптотическую сложность $O(N \log N)$.

Для решения исходной задачи в общем случае, когда каждая задача на межрегиональной олимпиаде оценивается своим количеством баллов, можно использовать различные подходы. Некоторые из них рассмотрены ниже в представленных вариантах решений.

Первый вариант решения**.** Он основан на использовании метода динамического программирования и заключается в следующем. Отсортируем все задачи, предложенные на межрегиональной олимпиаде, по *времени окончания* их решения. Пусть $a[i]$ – максимальное количество баллов, которые можно набрать, решая только задачи из числа "первых" i задач. Рассмотрим i-ю задачу. Пусть j – это номер последней задачи, которая заканчивается не позже, чем выдается i-я задача (то есть $s_j + t_j \le s_i$, а $s_{j+1} + t_{j+1} > s_j$). Тогда

$$a[i] = \max(a[i-1], a[j] + c[i]).$$

Из этого следует, что можно либо решить задачу с номером i и какой-то набор задач с номерами не больше j, либо не решать задачу с номером i. Ответом на поставленный вопрос будет число $a[n]$.

Осталось пояснить, как находить число j. Это можно делать несколькими способами:

- перебором всех чисел от 1 до i (решение с асимптотической сложностью $O(N^2)$);
- бинарным поиском на отрезке от 1 до i (сложность $O(N \log N)$);
- используя метод "двух указателей", то есть, запоминая предыдущее j и осуществляя поиск каждого следующего j на отрезке от предыдущего j, поскольку новое значение будет больше или равно предыдущему (таким образом, для вычисления всех $a[i]$ каждого кандидата каждый j мы проверим лишь единожды и сложность такого решения будет равна $O(N \log N)$ с учетом сортировки).

Восстановление ответа. Кроме поиска наибольшей суммы баллов, которую наберет Артур на олимпиаде, требуется также вывести количество и перечень задач, решение которых позволит ему набрать такую сумму. Это можно сделать одним из стандартных способов восстановления ответа в динамическом программировании. Например, будем кроме массива a хранить дополнительный массив *prev*, в котором укажем, какой выбор был сделан на каждом шаге. Если $a[i] = c[i] + a[j]$, то в $prev[j]$ запишем j. Это будет означать, что решается задача с номером i, а предыдущая задача имеет номер не больше j. В противном случае необходимо записать в $prev[i]$ число –1. Теперь, пройдя с конца по массиву *prev*, можно восстановить список решаемых задач.

Второй вариант решения. При реализации этого варианта вначале отсортируем вместе все начала и концы отрезков, сохраняя для каждой точки ее тип (начало или конец) и номер отрезка, к которому она относится. Будем рассматривать отсортированные события слева направо. В переменной *best* будем хранить лучший результат, который можно получить, используя только задачи, время выполнения которых уже закончилось к текущему моменту («закончившиеся» задачи). Если очередная точка – это начало отрезка с номером i, то запишем в $b[i]$ текущее значение переменной *best*. Таким образом, $b[i]$ – это наилучший результат, который можно получить до начала решения i-й задачи. Если очередная точка – это конец i-го отрезка, то необходимо обновить значение переменной *best* следующим образом: если решается i-я задача, то $best = b[i] + c[i]$, в противном случае значение переменной *best* не изменяется. Из двух вариантов нужно выбрать тот, в котором значение переменной *best* наибольшее, то есть $best = \max\{best, b[i] + c[i]\}$.

Задача 4 «Дом Мэра»

Если рассматривать небольшие ограничения, то решение задачи может быть следующим. Построим граф, соответствующий городу без застройки, при этом вершинам графа будут соответствовать перекрестки, а ребрам – дороги. Затем удалим в графе все ребра, попавшие в застроенные кварталы (сложность такой процедуры равна O(n * площадь города)). В полученном графе начнем поиск в ширину из точки (0, 0), чтобы определить расстояния до всех возможных расположений будущего дома Мэра, и выберем из них самое близкое. Восстанавливая путь при реализации поиска в ширину стандартным способом, можно найти точки поворота пути.

Если площадь города велика, а кварталов застройки не много, то можно воспользоваться *сжатием координат.* Для этого при построении графа будем использовать только те горизонтали и вертикали, на которых расположены мэрия, будущие дома или границы кварталов застройки.

В случае максимальных ограничений решение исходной задачи может быть следующим. Будем решать задачу отдельно для каждого возможного расположения будущего дома Мэра. Пусть рассматриваемый дом имеет координаты (x_i, y_i). Будем считать, что $x_i \geq 0$, $y_i \geq 0$. Легко понять, что нужно изменить в решении, чтобы рассмотреть и случаи отрицательных координат.

Рассмотрим пути, выходящие из мэрии на Север. Возможны два варианта пути:

1) едем на Север, затем поворачиваем направо (на Восток), и затем налево (опять на Север); при этом любой отрезок пути может иметь длину, равную 0;
2) едем на Север, пересекая горизонтальную улицу, на которой будет расположен дом Мэра, затем поворачиваем направо (на Восток), и затем еще раз направо (обратно на Юг).

Сравнивая эти два пути можно сказать, что первый путь всегда короче второго.

Далее выясним, как далеко мы сможем проехать из точки (0, 0) на Север (см. рис. 1). Для этого переберем все кварталы застройки и пересечем их лучом, выходящим из начала координат на Север. Пусть мы можем беспрепятственно проехать до точки $(0, R)$. Если рассматриваемый дом находится на этом отрезке, то задача решена.

Аналогично рассмотрим луч, выходящий из точки (x_i, y_i) на Юг, и найдем на нем точку (x_i, S) с минимальной положительной ординатой S, такую что из этой точки мы сможем беспрепятственно доехать до дома Мэра. Если $S > R$, то пути первого типа нет. В противном случае, попытаемся найти минимальное значение t на отрезке $[S, R]$, такое, что можно проехать из точки $(0, t)$ в точку (x_i, t). Для этого рассмотрим все кварталы, которые пересекаются с полосой $0 \le x \le x_i$. Рассмотрим их проекции на ось Oy, найдем объединение этих проекций (открытых интервалов), и найдем точку Y на оси Oy с минимальной координатой, не меньшей S, не покрытую объединением интервалов. Для этого можно отсортировать вместе начала и концы интервалов и пройти по этим точкам, считая *баланс*. Если Y окажется не больше R, то искомый путь имеет поворот направо в точке $(0, Y)$ и поворот налево в точке (x_i, Y). В противном случае, пути первого вида нет, и нужно перейти к поиску пути второго вида.

Если $R \le y_i$, то пути второго вида также нет. Иначе рассмотрим луч, выходящий из точки (x_i, y_i) на Север (см. рис. 2), и найдем на нем самую далекую точку S, до которой можно беспрепятственно добраться из точки (x_i, y_i). Рассмотрим отрезок $[x_i, \min\{R, S\}]$ и найдем на нем минимальную точку Y, не покрытую объединением проекций интервалов, рассмотренным выше. Если такая точка Y существует, то из всех путей, выходящих из точки (0, 0) на Север, кратчайший путь имеет повороты в точках $(0, Y)$ и (x_i, Y). В противном случае таких путей нет.

Заметим, что если $Y = y_i$, то второй поворот не нужен.

Данный алгоритм имеет асимптотическую сложность $O(kn \log n)$.

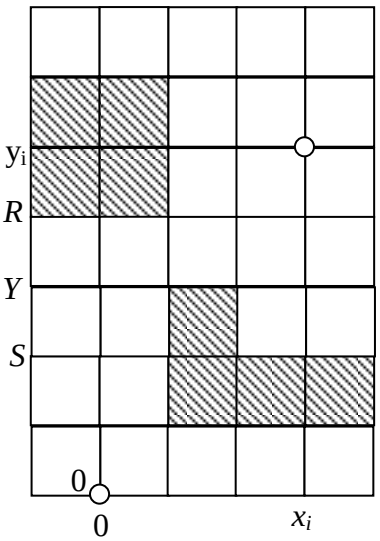


Рис. 1.

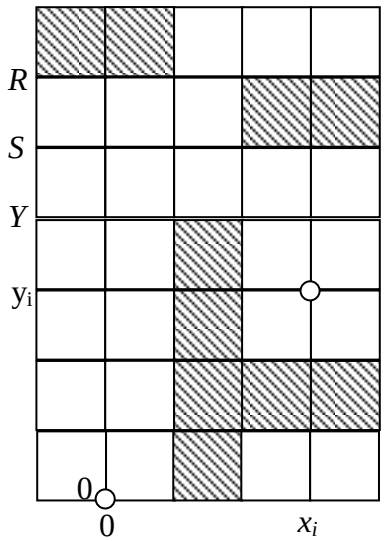


Рис. 2.

Задача 5 «Светофоры»

Поскольку решения с целочисленным ответом является частичными и также оцениваются, то начнем рассмотрение методов решения исходной задачи именно с таких решений. Для этого сформулируем следующую вспомогательную задачу: «*Электрокар движется со скоростью v. Может ли он проехать оба светофора на зеленый свет?*».

Для решения этой вспомогательной задачи введем в рассмотрение два электрокара А и Б. Пусть электрокар А проезжает первый светофор в момент включения зеленого сигнала, а электрокар Б – на a минут позже, то есть, в момент включения следующего красного сигнала. Ко второму светофору оба эти электрокара подъедут с разницей в a минут. Если хотя бы один из них окажется у второго светофора, когда у него будет гореть зеленый свет, то задача может быть легко решена. Если же оба электрокара подъедут к светофору с красным светом, то и никакой другой электрокар не сможет проехать на зеленый свет, потому что зеленый свет горит a минут и интервал между электрокарами А и Б также составляет a минут. Понятно, что за это время зеленый свет на светофоре не зажигался, и любой электрокар, стартовавший позже А, но раньше Б, подъедет ко второму светофору, когда у него будет гореть красный свет. Осталось теперь научиться определять, какого цвета будет второй светофор, когда к нему подъедут электрокары А и Б.

Время, необходимое электрокару, чтобы проехать от одного светофора до другого, равно x / v. Период работы светофора равен $(a + b)$. Поэтому необходимо, чтобы

выполнялись условия: $(x / v) \% (a + b) \le a$ (для первого электрокара) или $(x / v + a) \% (a + b) \le a$ (для второго электрокара). Эти условия также можно переписать, используя целочисленное деление. Например, для первого электрокара оно будет выглядеть следующим образом:

$$(x // v) \% (a + b) < a \text{ or } (x // v) \% (a + b) == a \text{ and } x \% v == 0$$

Теперь, когда понятно, как отвечать на вопрос, можно ли проехать оба светофора на зеленый свет, если двигаться со скоростью v, не трудно определить подходящую скорость среди целочисленных ответов. Для этого необходимо перебрать все целые скорости, начиная с 1000, до тех пор, пока не найдется правильный ответ.

В общем случае искомая скорость не обязательно является целочисленной. Для решения задачи в этом случае рассмотрим электрокары А и Б, скорости которых равны 1000 и которые проезжают первый светофор в момент включения и выключения у него зеленого света соответственно. Если один из них проезжает второй светофор на зеленый свет, то легко найти ответ, и задача решена. В противном случае, выгоднее всего будет уменьшить скорость электрокара Б так, чтобы он проехал второй светофор в момент включения зеленого света. Для этого он должен увеличить время своего движения $t = x / 1000$ на величину, равную:

$$dt = (a + b) – (x / 1000 + a) \% (a + b).$$

Тогда его скорость будет равна $x / (t + dt)$.

Задача 6 «Кондиционеры»

Начнем решение этой задачи с рассмотрения частичного решения. Это решение основано на переборе и заключается в следующем. Для каждого класса перебираются все кондиционеры, и выбирается из них подходящий кондиционер с минимальной стоимостью. Легко видеть, что асимптотическая сложность такого решения равна $O(n \cdot m)$.

Решение на полный балл будет выглядеть следующим образом. Заметим, что если одна модель кондиционера дороже другой, а при этом объем обслуживаемого им помещения не больше, чем у первого, то этот кондиционер покупать не выгодно. С учетом этого отсортируем все кондиционеры по стоимости, а при равной стоимости – по мощности. Далее, последовательно рассматривая полученные в этом случае элементы упорядоченного массива, удалим в нем все модели кондиционеров, которые имеют не большую мощность при равной или большей стоимости. Это можно сделать за линейное время, используя тот же массив, или создать новый массив. После этого получается

массив кондиционеров, в котором элементы расположены в порядке их возрастания как по стоимости, так и по мощности. Следовательно, для каждого класса нужно выбрать кондиционер из полученного массива с минимальной подходящей мощностью.

Теперь осталось упорядочить классы по требуемой мощности кондиционеров и пройтись по массиву классов и массиву кондиционеров «двумя указателями» - один указатель будет указывать на класс, другой на самый выгодный подходящий для него кондиционер. При увеличении мощности кондиционера для класса указатель, указывающий на самый выгодный кондиционер, будет только расти.

Такой алгоритм имеет сложность $O(n \cdot \log n + m \cdot \log m)$. Также можно осуществлять поиск кондиционера для каждого класса с использованием двоичного поиска. Сложность алгоритма в этом случае будет равна $O(m \cdot \log m + n \cdot \log m)$.

Задача 7 «Конфеты»

Для начала рассмотрим самый простой вариант решения данной задачи, когда $a = b = c = 1$. Не сложно увидеть, что в этом случае значения x, y и z должны быть по возможности равными, т.е. $x = y = z = n/3$. Если n не делится на 3, то какое-то из этих чисел надо округлить вверх, какое-то – вниз.

Для доказательства этого утверждения рассмотрим два случая: первый, когда $x = y = z = n/3$, и второй, когда $x = n/3 + t$, $y = n/3 – t$, $z = n/3$. В первом случае $xyz = (n^3/27)$, во втором $xyz = (n^3/27) – t^2n/3$, что меньше при любом ненулевом значении t.

В общем случае можно предположить, что ax, by, cz близки к $n/3$. Чтобы убедиться в этом, сначала исследуем это утверждение для более простой задачи: $ax + by = n$.

Рассмотрим начальное значение $x = \left\lfloor \frac{n}{2a} \right\rfloor$ и $y = \left\lfloor \frac{n}{2b} \right\rfloor$, то есть, такие x и y, что ax и by близки к $n/2$. Заметим, что $ax > \frac{n}{2} - a$ и $by > \frac{n}{2} - b$. Поэтому $ax \times by = f_0 > (\frac{n}{2} - a) \times (\frac{n}{2} - b) = (\frac{n}{2})^2 - (a+b)(\frac{n}{2})$. Попробуем теперь взять значения x и y, отличающиеся от $n/2$ на t и исследуем, при каких значениях t результат может быть лучше. Получаем $ax \times by = f_t > (\frac{n}{2} - t) \times (\frac{n}{2} - t) = (\frac{n}{2})^2 - t^2$. Если $t^2 > (a+b)\frac{n}{2}$, то этот результат меньше, чем при первых значениях x и y.

Рассмотрим, при каких значениях t результат может быть лучше. Пусть $t = a \times dx = b \times dy$ и $a \geq b$. Тогда из предыдущего неравенства получается, что интересны

$t^2 \le (a + b)\frac{n}{2} \le an$, т.е. $(a \times dx)^2 \le an \Leftrightarrow a \times dx^2 \le n$. Заметим, что $dx \le b \le a$. Получается, что $dx^3 \le n$. Значит, лучший результат может быть только при $dx \le \sqrt[3]{n}$.

Осталось теперь обобщить идею для трех переменных. Заметим, что для любых двух из трех неизвестных можно провести такое же рассуждение, то есть, *ax* и *by* не могут отличаться больше, чем на кубический корень из *n*, а также пара *ax* и *cz* и пара *by* и *cz* не могут сильно отличаться.

С учетом сказанного решение исходной задачи будет следующим.

1. Берем в качестве первого приближения $x = \left\lfloor \frac{n}{3a} \right\rfloor$, $y = \left\lfloor \frac{n}{3b} \right\rfloor$ и $z = \left\lfloor \frac{n}{3c} \right\rfloor$.

2. Переберем все перестановки *a, b* и *c*.

3. Переберем *x* в интервале от $\left\lfloor \frac{n}{3a} \right\rfloor - \sqrt[3]{n}$ до $\left\lfloor \frac{n}{3a} \right\rfloor + \sqrt[3]{n}$

4. Переберем *y* в интервале от $\left\lfloor \frac{n - ax}{2b} \right\rfloor - \sqrt[3]{n}$ до $\left\lfloor \frac{n - ax}{2b} \right\rfloor + \sqrt[3]{n}$

5. Получаем $z = \left\lfloor \frac{n - ax - by}{c} \right\rfloor$ и проверяем, стало ли произведение больше текущего.

Такое решение имеет асимптотическую сложность O($n^{2/3}$). Заметим, что количество коробок конфет, определяемое произведением размеров коробки, может быть порядка 10^{27}, то есть, превышать максимальное значение целого 64-битного типа данных. Для того, чтобы избежать длинной арифметики при сравнении перебираемых вариантов, можно вместо неравенства вида $x_1 \times y_1 \times z_1 > x_2 \times y_2 \times z_2$ использовать неравенство $x_1 \times y_1 \times z_1 - x_2 \times y_2 \times z_2 > 0$. Если разность не превышает 2^{63}, то неравенство с переполнением будет определено верно. Также можно использовать вещественный тип с расширенной точностью.

Следует отметить, что частичное решение, основанное на переборе всевозможных значений *x* и *y* и вычислении *z* по формуле $z = \left\lfloor \frac{n - ax - by}{c} \right\rfloor$, имеет асимптотическую сложность O(n^2) и оценивается из 40 баллов.

Задача 8. «Волонтеры»

После формализации исходную задачу можно сформулировать в терминах теории графов следующим образом. Пусть заданы два дерева с общими листьями (рис. 1).

Требуется найти количество пар вершин (*u, v*), где *u* принадлежит левому дереву, а *v* – правому, для которых существует путь из вершины *u* в вершину *v*, идущий всё время слева направо, как изображено на рис. 2.

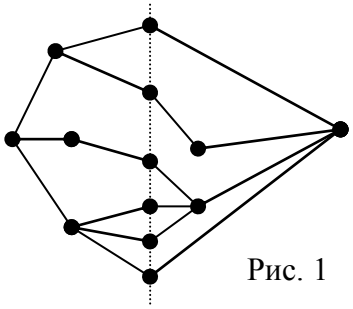


Рис. 1

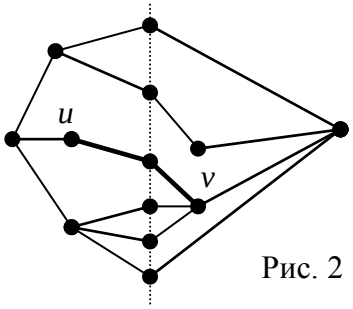


Рис. 2

Пронумеруем листья деревьев, и для каждой из остальных вершин определим, какие листья ей соответствуют (см. рис. 3). Заметим, что каждой вершине соответствуют несколько последовательных листьев, поэтому достаточно хранить только номера самого нижнего и самого верхнего листа. Это можно сделать, реализовав в каждом дереве из корня обход в глубину и отметив для каждой вершины номер самого первого потомка-листа и самого последнего потомка-листа. Суммарная асимптотическая сложность такой реализации будет равна O($n + m + k$).

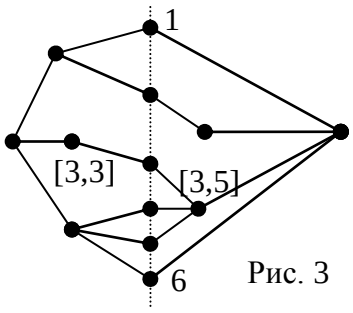


Рис. 3

Далее можно просто рассмотреть каждую пару вершин (*u, v*) и определить, имеют ли отнесенные к ним отрезки общее пересечение. Реализующий эти операции алгоритм будет иметь сложность O($m \cdot k$). Более эффективное решение можно построить следующим образом.

Пусть отрезки, отнесенные к вершинам левого дерева, будут синими, а отрезки, отнесенные к вершинам правого дерева, – красными. Отметим эти отрезки на одной координатной прямой. Подсчитаем для каждого синего отрезка, с каким количеством красных отрезков он пересекается. Это можно сделать следующим образом.

Будем двигаться по координатной прямой слева направо и подсчитывать отдельно количество начал красных отрезков и количество концов красных отрезков. Если рассмотреть конкретный синий отрезок AB, то можно установить, что с ним пересекаются

все красные отрезки, кроме тех, которые заканчиваются левее A, и тех, которые начинаются правее B. Таким образом, дойдя до вершины A, мы определим количество концов красных отрезков, которые нам встретились, а дойдя до вершины B – количество начал красных отрезков, которые нам еще не встретились. Зная эти числа, можно легко вычислить искомое количество отрезков. Асимптотическая сложность такого решения будет равна $O((m + k)\cdot\log(m + k))$, включая сортировку концов отрезков.

Существует еще один способ реализации похожей идеи. Он заключается в следующем. Будем подсчитывать количество пар (u, v), таких, что из вершины u в вершину v нет пути. Для этого сначала подсчитаем для вершин *левого* дерева те же отрезки листьев, что и в предыдущем способе решения. С вершинами правого дерева поступим иначе. Для каждого листа w рассмотрим путь от него до корня правого поддерева (см. рис. 4) и подсчитаем отдельно количество вершин выше этого пути (назовем его $\text{top}[w]$) и ниже этого пути (назовем его $\text{bottom}[w]$).

Теперь рассмотрим вершину левого дерева. Научимся за O(1) определять, со сколькими вершинами правого поддерева она *не* соединена. Для этого рассмотрим отрезок $[a, b]$, который ей соответствует. Ясно, что она не соединена с $(\text{top}[a] + \text{bottom}[b])$ вершинами правого поддерева. Осталось просуммировать данные выражения для всех вершин левого поддерева.

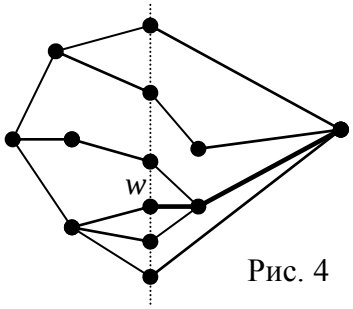


Рис. 4

Теперь, чтобы получить ответ на поставленную исходную задачу, достаточно вычесть полученное выше количество пар из общего количества пар $m\cdot k$.

6. Список рекомендуемой литературы

1. Алексеев В.Е., Таланов В.А. Графы и алгоритмы. Структуры данных. Модели вычислений. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006. – 320 с. – (Серия «Основы информационных технологий»)
2. Арсак Ж. Программирование игр и головоломок. – М.: Наука, 1990. – 224 с.
3. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Издательский дом «Вильямс», 2000. – 384 с.
4. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. — Пер. с англ. — М.: Мир, 1979. — 536 с.
5. Бентли Д. Жемчужины творчества программистов: пер. с англ. – М.: Радио и связь, 1990. – 224 с.
6. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1985.
7. Вирт Н. Алгоритмы и структуры данных. Пер. с англ. М.: Мир, 1989. – 360 с.
8. Гасфилд Дэн. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология / Пер. с англ. И.В.Романовского. – СПб.: Невский Диалект; БХВ Петербург, 2003. – 654 с.
9. Джонстон Г. Учитесь программировать. – М.: Финансы и статистика, 1989. – 336 с.
10. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. – М.: Наука, 1990. – 384 с.
11. Задачи по программированию /С.М. Окулов, Т.В. Ашихмина, Н.А. Бушмелева и др.; Под ред. С.М. Окулова. – М.: БИНОМ. Лаборатория знаний, 2006. – 820 с.
12. Златопольский Д. М. Программирование: типовые задачи, алгоритмы, методы. – М.: БИНОМ. Лаборатория знаний, 2007. – 223 с.
13. Кирюхин В.М. Методика проведения и подготовки к участию в олимпиадах по информатике. Всероссийская олимпиада школьников. – М.: БИНОМ. Лаборатория знаний, 2011. – 271 с.
14. Кирюхин В.М. Информатика. Всероссийские олимпиады. Выпуск 1. – М.: Просвещение, 2008. – 220 с. – (Пять колец).
15. Кирюхин В.М. Информатика. Всероссийские олимпиады. Выпуск 2. – М.: Просвещение, 2009. – 222 с. – (Пять колец).

16. Кирюхин В.М. Информатика. Всероссийские олимпиады. Выпуск 3. – М.: Просвещение, 2011. – 222 с. – (Пять колец).
17. Кирюхин В.М. Информатика. Всероссийские олимпиады. Выпуск 4. – М.: Просвещение, 2013. – 222 с. – (Пять колец).
18. Кирюхин В.М. Информатика. Международные олимпиады. Выпуск 1. – М.: Просвещение, 2009. – 239 с. – (Пять колец).
19. Кирюхин В.М., Окулов С. М. Методика решения задач по информатике. Международные олимпиады. – М.: БИНОМ. Лаборатория знаний, 2007. – 600 с.
20. Кнут Д. Искусство программирования для ЭВМ. Т. 1-3. – М., СПб., Киев: Вильямс, 2000.
21. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 1999. – 960с.
22. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.
23. Липский В. Комбинаторика для программистов. – М.: Мир, 1988. – 77 с.
24. Майерс Г. Искусство тестирования программ. Пер. с анг. под ред. Б.А. Позина. – М.: Финансы и статистика, 1982. – 176 с.
25. Никулин Е.А. Компьютерная геометрия и алгоритмы машинной графики. – СПб.: БХВ-Петербург, 2003. – 560 с.
26. Окулов С.М. Программирование в алгоритмах. – М.: БИНОМ. Лаборатория знаний. 2002. – 341 с.
27. Окулов С.М. Дискретная математика. Теория и практика решения задач по информатике: учебное пособие. – М.: БИНОМ. Лаборатория знаний. 2008. – 422 с.
28. Окулов С.М. Алгоритмы обработки строк: учебное пособие. – М.: БИНОМ. Лаборатория знаний, 2009. – 255 с.
29. Окулов С.М. Абстрактные типы данных. – М.: БИНОМ. Лаборатория знаний, 2009. – 250 с.
30. Окулов С.М. Динамическое программирование. – М.: БИНОМ. Лаборатория знаний, 2011. – 260 с.
31. Просветов Г.И. Дискретная математика: задачи и решения: учебное пособие. – М.: БИНОМ. Лаборатория знаний. 2008. – 222 с.
32. Рейнгольд Э. Комбинаторные алгоритмы: теория и практика / Э. Рейнгольд, Ю. Нивергельт, Н. Део. – М.: Мир, 1980. – 476 с.

33. Столяр С.Е., Владыкин А.А.. Информатика. Представление данных и алгоритмы. – СПб.: Невский Диалект; М.: БИНОМ. Лаборатория знаний. 2007. –382 с.
34. Уэзерелл Ч. Этюды для программистов. – М.: Мир, 1982. – 288 с.
35. Шень А. Программирование: теоремы и задачи. – М.:МЦНМО, 1995. – 264 с.