



В. И. Галочкин

Поволжский государственный технологический университет, г. Йошкар-Ола, Республика Марий Эл, Россия

ЗАДАЧИ ПО ПРОГРАММИРОВАНИЮ ИНТЕЛЛЕКТУАЛЬНОЙ ОЛИМПИАДЫ 2020 ПРИВОЛЖСКОГО ФЕДЕРАЛЬНОГО ОКРУГА СРЕДИ ШКОЛЬНИКОВ

Аннотация

В статье рассмотрены результаты ежегодной Интеллектуальной олимпиады по программированию Приволжского федерального округа среди школьников, в которой участвовали представители четырнадцати регионов. Каждая команда состояла из двух школьников, прошедших отбор в региональных турнирах. Приведены авторские решения задач олимпиады. Набор задач сбалансирован по тематике и трудности. Темы задач связаны с организацией рациональных структур данных, скобочными последовательностями, оптимизацией на графах, поиском в глубину и ширину, анализом областей на клетчатом поле, построением и оценкой дерева игры. Важными условиями успешного выступления оказались хорошее командное взаимодействие и правильная оценка вычислительной сложности алгоритмов. Победителем олимпиады стала команда Самарской области, решившая все восемь предложенных задач. Статья предназначена для школьников, студентов, а также для тренеров, ведущих занятия по олимпиадному программированию.

Ключевые слова: программирование, информатика, олимпиада, задача, алгоритм, программа.

DOI: 10.32517/2221-1993-2020-19-4-36-43

Введение

В марте 2019 года в городе Йошкар-Ола на базе Поволжского государственного технологического университета прошел окружной этап Интеллектуальной олимпиады Приволжского федерального округа среди школьников. Организация и содержательная часть олимпиады были признаны успешными, поэтому нынешний этап в марте 2020 года снова состоялся в Йошкар-Оле.

Как и в прошлом году, в олимпиаде по программированию приняли участие команды четырнадцати регионов Приволжского федерального округа. Каждая команда состояла из двух школьников, прошедших отбор в региональных отборочных турнирах.

Команде выделялся один компьютер. Для проверки решений использовалась распространенная система Ejudge [10]. Полное решение каждой задачи оценивалось 100 баллами. Частичные решения получали оценки, пропорциональные количеству пройденных тестов. Такой

подход в целом соответствует правилам всероссийской олимпиады школьников [3].

Автором статьи составлены все восемь задач разной тематики и сложности, разработаны алгоритмы и программы, подготовлены тесты. Эти материалы были приняты жюри без изменений.

Для всех задач, кроме A и C, время работы программы на одном тесте не должно превышать одной секунды, а затраты оперативной памяти — 256 мегабайтов. Задачи A и C имели ограничение по времени — две секунды.

Задача A. Великолепные четверки

Имеется последовательность натуральных (целых положительных) чисел $x_1, x_2, \dots, x_n$ ($4 \leq n \leq 2000$, $1 \leq x_i \leq 10^{15}$). Великолепной четверкой называется набор из четырех чисел (x_i, x_j, x_k, x_m) , в котором все индексы попарно различны и одно из чисел равно сумме трех других, т. е. $x_i + x_j + x_k = x_m$. Две четверки считаются

Контактная информация

Галочкин Владимир Иванович, канд. тех. наук, доцент, доцент кафедры информатики и системного программирования, Поволжский государственный технологический университет, г. Йошкар-Ола, Республика Марий Эл, Россия; адрес: 424000, Россия, Республика Марий Эл, г. Йошкар-Ола, пл. Ленина, д. 3; e-mail: info@volgatech.net

V. I. Galochkin

Volga State University of Technology, Yoshkar-Ola, The Mari El Republic, Russia

TASKS FOR PROGRAMMING OF THE INTELLECTUAL OLYMPIAD OF 2020 OF THE VOLGA FEDERAL DISTRICT AMONG SCHOOLCHILDREN

Abstract

The article presents the results of the annual Intellectual Olympiad for programming of the Volga Federal District among schoolchildren, in which representatives of fourteen regions participated. Each team consisted of two schoolchildren who were selected in regional contests. The author's solutions of the Olympiad tasks are considered. The set of tasks is balanced by subject and difficulty. The topics of the tasks are related to the organization of rational data structures, bracket sequences, optimization in graphs, depth-first and breadth-first search, analysis of areas on a checkered field, construction and evaluation of the game tree. Important conditions for a successful performance were good teamwork and a correct estimation of the computational complexity of the algorithms. The winner of the Olympiad was the team of the Samara region, which solved all eight of the proposed tasks. The article is intended for schoolchildren, students, trainers, leading classes in Olympiad programming.

Keywords: programming, informatics, Olympiad, task, algorithm, program.

разными, если множества индексов отличаются хотя бы одним элементом. Требуется найти количество велико-лепных четверок.

Входные данные.

В первой строке содержится значение n .

Во второй строке заданы через пробел значения $x_1, x_2, \dots, x_n$.

Выходные данные.

Вывести количество указанных четверок чисел.

Примеры.

№ п/п	Ввод	Вывод	Комментарий
1	4 2 6 1 3	1	
2	7 3 2 6 3 4 7 1	4	Четверки $x_i + x_j + x_k = x_m$ образуют числа с индексами (1, 2, 7, 3), (2, 4, 7, 3), (2, 5, 7, 6), (1, 4, 7, 6).

Описание алгоритма.

Отсортируем по неубыванию исходный массив и будем считать, что $x_1 \leq x_2 \leq \dots \leq x_n$. Преобразуем равенство $x_i + x_j + x_k = x_m$ к виду $x_i + x_j = x_m - x_k$ при $m > k$ и сформируем два новых массива: $y_1, y_2, \dots, y_s$ и $z_1, z_2, \dots, z_s$ при $s = n \cdot (n - 1) / 2$ из попарных сумм и разностей, соответствующих левой и правой частям последнего равенства.

Отсортируем оба массива по неубыванию и будем связанно продвигаться по ним, переходя вперед по массиву с меньшим текущим элементом. Если $y_i = z_j$, а значения y_i и z_j повторяются a и b раз соответственно, то в счетчик количества четверок добавится произведение $a \times b$.

Будет ли полученное количество четверок ответом? Нет!

Во-первых, ошибочно учтен вариант, когда $x_i + x_j = x_m - x_i$ или $2x_i + x_j = x_m$, а также симметричный вариант $x_i + 2x_j = x_m$. Сформируем массив $u_1, u_2, \dots, u_s$ из значений $2x_i + x_j$ и отсортируем его по неубыванию. Продвигаясь связанно по этому и исходному массивам, определим, как уже описано, количество пар (p, q) , когда $u_p = x_q$. Это количество нужно вычесть из ранее найденного числа четверок. Аналогично определим, сколько раз выполняется соотношение $x_i + 2x_j = x_m$, и снова уменьшим количество четверок.

Во-вторых, из равенства $x_i + x_j + x_k = x_m$ следуют три соотношения:

$$x_i + x_j = x_m - x_k,$$

$$x_i + x_k = x_m - x_j$$

$$\text{и } x_j + x_k = x_m - x_i.$$

Поэтому четверка чисел (x_i, x_j, x_k, x_m) будет учтена трижды. Следовательно, полученное количество четверок нужно поделить на 3.

Сложность алгоритма составляет $O(n^2 \log n)$.

Задача В. Циклический сдвиг

Последовательность круглых скобок называют правильной, если можно добавить в нее числа и знаки арифметических операций так, что полученное алгебраическое выражение станет корректным. Например,

скобочная последовательность $((()))$ является правильной, а последовательности $((()))$ и $((()))$ правильными не являются.

Задана скобочная последовательность из n скобок $a_1, a_2, \dots, a_n$ ($1 \leq n \leq 6 \cdot 10^5$). Циклическим сдвигом называется операция, результатом которой является циклическая последовательность $a_i, a_{i+1}, \dots, a_n, a_1, a_2, \dots, a_{i-1}$ ($1 \leq i \leq n$).

Найти минимальный номер позиции, начиная с которой циклическая последовательность окажется правильной.

Входные данные.

В первой строке задано значение n .

Во второй строке находится скобочная последовательность длины n .

Выходные данные.

Вывести номер позиции.

Если правильную скобочную последовательность получить невозможно, вывести No.

Примеры.

№ п/п	Ввод	Вывод
1	6 ()))((	5
2	6 ((()((	No

Описание алгоритма.

Правильность последовательности определяется расчетом для каждой позиции глубины s вложенности скобок. Положим $s = 0$ и пройдем по последовательности. При встрече открывающей скобки будем увеличивать, а при встрече закрывающей скобки — уменьшать значение s на единицу. Сохраним минимальное значение s в переменной m .

При $s \neq 0$ количество открывающих скобок не совпадает с количеством закрывающих, поэтому ответом будет «No». Если в конце просмотра окажется, что $s = 0$ и $m = 0$, то последовательность правильная и ответ 1.

Пусть сейчас $s = 0$ и $m < 0$. Обозначим через p первую позицию, где достигнуто минимальное значение m . Для последней позиции $s = 0$, поэтому $p < n$. В позиции $p + 1$ должна быть открывающая скобка. Иначе глубина скобок в позиции $p + 1$ будет на 1 меньше m . Значит, в позиции $p + 1$ глубина $s = m + 1$.

Начнем снова расчет глубины s от позиции $p + 1$ с переходом на начало. До конца исходной последовательности прежние значения s увеличатся на $-m$, т. е. не будут отрицательными. Прежнее значение s в первой позиции было равно 1 в случае открывающей скобки либо -1 — в случае закрывающей скобки. Новое значение s в конце строки при циклическом переходе к началу либо увеличится, либо уменьшится на 1. Поскольку в последней позиции оно превосходило на величину $-m$ прежнее значение, то как в начале строки, так и далее сохранится та же самая разность $-m$. В частности, в позиции p новое значение s станет равным 0.

Итак, в циклической последовательности с позиции $p + 1$ глубина вложенности скобок s не принимает от-

рицательные значения и в конце строки обращается в 0, поэтому циклическая последовательность от позиции $p + 1$ является правильной. Значит, при $c = 0$ и $m < 0$ ответ дает значение $p + 1$.

Сложность алгоритма оценивается величиной $O(n)$.

Задача С. Электрички

Железная дорога имеет n станций, пронумерованных в порядке их расположения от 1 до n ($2 \leq n \leq 800$). Электрички курсируют в обоих направлениях. Каждая из m электричек ($1 \leq m \leq 800$) может следовать между любыми двумя станциями, останавливаясь как на всех промежуточных станциях, так и на некоторых из них. Электрички тратят одинаковое время t_i на преодоление расстояния между соседними станциями i и $i + 1$ независимо от своих остановок и направления движения ($1 \leq i \leq n - 1$, $1 \leq t_i \leq 10^6$). Известно расписание движения электричек. Пассажир Вася в определенный момент времени h ($1 \leq h \leq 10^6$) прибыл на станцию с номером a . Ему нужно как можно раньше добраться до станции b . Помогите Васе.

Входные данные.

В первой строке задано число станций n , количество электричек m , номера станций a и b и момент прибытия h на станцию a .

Во второй строке содержатся $n - 1$ значений $t_1, t_2, \dots, t_{n-1}$.

В следующих m строках задается расписание движения поездов. Для каждого i -го поезда указываются время отправления s_i от начальной станции, количество остановок k_i , включая начальную и конечную станции, номера станций для остановки в порядке следования электрички.

Все числа целые и не превышают 10^6 .

Выходные данные.

Вывести наиболее ранний момент времени достижения станции b .

Если станция b недостижима, вывести No.

Время стоянки электричек на станциях не учитывается.

Примеры.

№ п/п	Ввод	Вывод	Комментарий
1	4 2 3 4 2 3 4 5 7 3 3 2 1 7 3 1 2 4	20	На первой электричке Вася проезжает в интервал времени (7, 11) от станции 3 до станции 2. Вторая электричка прибывает на станцию 2 в момент времени 11. Вася пересаживается на нее и попадает на станцию 4 в момент времени 20
2	4 2 3 4 2 3 4 5 7 3 3 2 1 8 3 1 2 4	No	Вася на станции 2 не успевает на вторую электричку, ушедшую ранее в момент 10

Описание алгоритма.

Решение задачи реализуется поиском в ширину в стиле алгоритма Дейкстры [4, 6, 8, 9, 11]. Для каждой станции будем вычислять временные и постоянные метки,

определяя момент наиболее раннего прибытия на данную станцию. Первоначально станция a получает постоянную метку h , а остальные станции — временную метку ∞ .

На каждом шаге будем от станции, получившей постоянную метку последней, просматривать в цикле все поезда, у которых время прибытия на эту станцию такое же или более позднее. У этих поездов определим время прибытия на все станции, где они останавливаются. Для каждой станции находим минимальное время между вновь определенным и уже имеющимся. Станция с минимальным временем прибытия получает постоянную метку, равную наиболее раннему возможному времени прибытия пассажира на станцию.

Вычисления прекращаются, если станция b получит постоянную метку либо не уменьшается ни один из моментов наиболее раннего прибытия на какую-либо станцию.

Имеет смысл заранее при вводе информации вычислить время прохождения r_i до каждой i -й станции от начальной станции 1. Тогда отрезок от i -й до j -й станции в любом направлении преодолевается за время $|r_j - r_i|$.

Сложность алгоритма составляет $O(mn^2)$, но малый объем вычислений на каждом шаге позволяет уложиться в отведенное время.

Задача D. Дамба

В рыбхозе «Рыбнастол» принято решение о разведении карасей и шук. К сожалению, эти рыбы не могут быть вместе в одном водоеме, так как шуки предпочитают питаться исключительно карасями. Планируется каждое из k озер ($1 \leq k \leq 10$) разделить дамбой на две не сообщающиеся друг с другом части.

На карте каждое i -е озеро представлено в определенном масштабе прямоугольным участком $m_i \times n_i$ единиц ($1 \leq m_i, n_i \leq 1000$), разбитым на единичные квадраты. Отдельный квадрат целиком занят водой или суши. Множество водных квадратов является связным. Это означает, что из любого водного квадрата озера можно попасть в любой другой водный квадрат, последовательно переходя по водным квадратам через их общую сторону. Такие последовательности связанных квадратов будем называть путями.

Дамба имеет вид непрерывной ломаной, проходящей по сторонам квадратов. Два водных квадрата, общая сторона которых принадлежит ломаной, становятся не сообщающимися напрямую друг с другом. Требуется для каждого озера определить минимальную по количеству сторон длину дамбы, разделяющей озеро на две не сообщающиеся между собой связные части.

Чтобы обеспечить доступ рыбаков к воде, каждая из двух полученных частей озера должна сообщаться с берегом. Иными словами, каждая часть должна содержать водный квадрат, который либо сам находится на границе исходного прямоугольного участка, либо имеет общую сторону с квадратом суши, связанным с границей путем из квадратов суши.

Входные данные.

В первой строке содержится число k .

В следующих строках следуют k блоков данных. Каждый блок описывает одно озеро.

В первой строке блока содержатся числа m_i и n_i , разделенные пробелом.

В следующих m_i строках находится матрица, представляющая озеро, по n_i символов в строке. Символ '.' обозначает квадрат суши, а символ '#' — квадрат воды. Гарантируется наличие не менее двух водных квадратов.

Общая площадь озер $s = m_1 \times n_1 + m_2 \times n_2 + \dots + m_k \times n_k$ не должна превосходить 10^6 .

Выходные данные.

В единственной строке вывести через пробел k значений, определяющих минимальные длины дамб.

В результате каждое озеро должно быть разделено на две части так, что водные клетки из разных частей не могут иметь общей стороны, не принадлежащей дамбе. Каждая часть должна быть связана с берегом так, как это определялось выше.

Пример.

Ввод	Вывод
<pre>2 2 6 .####. .####. 3 5 ##.## ##### ##.##</pre>	<pre>2 1</pre>

Описание алгоритма.

Отметим сразу, что в одном озере можно построить дамбу длиной не более 2. Действительно, выберем из самых левых водных квадратов верхний. Нижняя и правая стороны квадрата всегда отгораживают его от остальной части озера. Значит, нужно попытаться найти дамбу единичной длины.

Рассмотрим возможные подходы к решению. Водным квадратам озера соответствует связный неориентированный граф. Вершинами графа являются квадраты, а ребра определяют связи с соседними водными квадратами. Можно определить, имеются ли мосты, т. е. ребра, удаление которых разделяет граф на два связных подграфа. Известен алгоритм нахождения мостов на основании поиска в глубину (DFS), трудоемкость которого составляет $O(V + E)$, где V и E — количество вершин и ребер графа соответственно [4, 6]. Тем не менее такой подход требует проверки дополнительных условий, в чем убеждает следующий пример исходных данных:

```
4 5
#####
##.##
#.##
#####
```

Имеющийся мост между квадратами с координатами (3, 3) и (4, 3) выделяет часть озера из одного квадрата (3, 3), не связанного с границей всего участка.

Опишем другой подход.

Организуем барьер, расширив заданный прямоугольник со всех четырех сторон квадратами суши. Пометим метками первого типа внешность озера из квадратов суши, используя процедуру заливки 4-связной области с рекурсивными переходами к четырем соседним квадратам [5].

Затем аналогично присвоим метки второго типа квадратам 8-связной внешности, считая, что соприкасающиеся углами квадраты также являются соседними. При этом часть квадратов, имеющих метку первого типа, получают также метки второго типа. Будем искать перебором разделяющую водные квадраты сторону, каждый конец которой является углом внешнего квадрата с меткой второго типа. Такая сторона разделяет озеро на две несвязанные части, однако наличие пути от каждой части до границы области не гарантируется. Так, в примере:

```
6 6
####.#
###.##
##.##
#.####
##.##
#####
```

сторона между квадратами (4, 4) и (4, 5) выделяет область из трех водных квадратов, не имеющих путей к границе области.

Рассматривая всевозможные варианты, можно доказать, что если хотя бы один из концевых квадратов стороны, разделяющей озеро, помимо меток второго типа имеет также метку первого типа, то дамба из этой стороны удовлетворяет условию. Поэтому задача сводится к перебору сторон, разделяющих водные квадраты, и анализу их концов. Если необходимая сторона находится, то у соответствующего озера минимальная длина дамбы 1, в противном случае — 2.

Сложность алгоритма составляет $O(s)$.

Задача Е. Магические квадраты

Вася учится в первом классе школы магов. Однажды Главный Магистр рассказал о магических квадратах. Такой квадрат порядка n состоит из n строк по n целых чисел в строке и должен иметь одинаковую сумму чисел в строках, столбцах и на длинных диагоналях. Вася попросил показать, как строятся такие квадраты.

Главный Магистр сказал, что это будет проходиться в следующей четверти, а пока стоит научиться строить полумагические квадраты. В них одинаковая сумма чисел s должна быть в строках и столбцах, но на диагоналях сумму s получать не обязательно. Вася много раз заполнял квадрат начальными числами, но дальше часто что-то не клеилось. Нужно срочно ему помочь. Значения n и s удовлетворяют ограничениям: $1 \leq n \leq 1000$, $1 \leq s \leq 10^6$.

Входные данные.

В первой строке содержатся числа n и s , разделенные пробелом.

В следующих n строках задаются значения матрицы a_{ij} по n целых чисел в строке через пробел ($0 \leq a_{ij} \leq 1000$), где 0 обозначает пустую клетку. Требуется заполнить пустые клетки целыми числами так, чтобы сумма чисел в каждой строке и в каждом столбце оказалась равной s .

Выходные данные.

В n строках по n чисел в строке вывести через пробел элементы построенного квадрата b_{ij} ($-10^9 \leq b_{ij} \leq 10^9$). В клетках могут содержаться как положительные значения, так и отрицательные либо нулевые.

Если ответов больше одного, вывести любой из них.
Если решения нет, вывести No.

Примеры.

№ п/п	Ввод	Вывод
1	3 85 0 0 43 25 0 0 0 5 18	-2 44 43 25 36 24 62 5 18
2	2 21 0 9 6 0	No
3	2 5 0 0 0 0	5 0 0 5

Описание алгоритма.

Назовем для удобства строки и столбцы квадрата линиями. Каждая клетка находится на пересечении двух линий, которые будем называть связанными.

Выберем одну из линий с минимальным числом свободных клеток. Если она не имеет свободных клеток, то проверим сумму находящихся в ней чисел. Если эта сумма не совпадает с s , то выдается отрицательный ответ «No». В противном случае определим следующую минимальную по числу незаполненных клеток линию. Этот процесс может закончиться либо получением отрицательного ответа, либо перебором всех линий с последней выдачей исходного квадрата, если он уже является полумагическим, либо нахождением линии с пустыми клетками. Рассмотрим последний случай.

Будем последовательно заполнять свободные клетки выбранной линии, уменьшая при этом количество свободных клеток в связанных линиях. В каждую свободную клетку, если она не последняя в линии, будем заносить какое-либо определенное значение, например 1. Последнее в линии значение однозначно определяется предыдущими значениями.

Продолжим в цикле описанные действия, выбирая для заполнения очередную линию с минимальным числом свободных клеток. Ею снова может оказаться уже заполненная линия, которая может образоваться в результате выполнения предыдущих шагов. Вычисления закончатся либо заполнением всех клеток квадрата, либо получением отрицательного ответа. Легко доказать, что если все линии формировались правильно и при заполнении последней пустой клетки квадрата обеспечивается правильная сумма по одной из линий, то и связанная по этой клетке линия будет иметь ту же сумму, т. е. квадрат окажется полумагическим.

Всегда ли описанный процесс приведет к получению решения, если оно существует? Да, всегда!

Действительно, пусть некоторое решение существует и квадрат может быть заполнен значениями b_{ij} . Пусть, например, описанный выше процесс начинается с занесения значения 1 в пустую клетку (i_1, j_1) строки i_1 . Попытаемся изменить некоторые значения b_{ij} в первоначально пустых клетках так, чтобы квадрат остался полумагическим.

Если в строке i_1 останется единственная пустая клетка, то закончим заполнение этой строки. Если это не так, уменьшим на 1 значение одной из пустых клеток

(i_1, j_2) этой строки. Общая сумма чисел в строке i_1 не изменится, но в столбце j_2 сумма уменьшится на 1. Мы заполняли линию с минимальным количеством пустых клеток, поэтому сейчас уже в столбце j_2 найдется пустая клетка (i_2, j_2) . Увеличим ее значение на 1 и перейдем к рассмотрению строки i_2 .

Может оказаться, что из последовательности $i_1, j_1, j_2, i_2, j_3, i_3, \dots$ выделится циклическая подпоследовательность типа $i_k, j_k, j_{k+1}, i_{k+1}, \dots, i_k$ из четного числа элементов. Это означает, что после описанного изменения чисел в клетках в соответствующих линиях суммы чисел останутся прежними.

В конце концов мы можем либо прийти к пустой клетке столбца j_1 , либо оказаться на линии с единственной пустой клеткой. В первом случае суммы чисел во всех линиях не изменятся, а во втором остается закончить заполнение соответствующей линии.

Рассуждая аналогично, можно подтвердить обоснованность занесения 1 либо другого значения в пустые клетки и на последующих шагах. Впрочем, участникам олимпиады не требовалось доказывать корректность подобной стратегии, а интуитивно она понятна.

Для эффективной реализации алгоритма удобно использовать бинарную кучу по минимуму [4, 8]. Занесем в нее номера строк и столбцов так, чтобы в вершине оказалась линия с минимальным числом свободных клеток. На каждом шаге число свободных клеток уменьшается, но линия остается в вершине до своего окончательного заполнения. Затем эта линия удаляется из кучи, а в вершину помещается новая минимально свободная линия.

Общая сложность алгоритма оценивается величиной $O(n^2 \log n)$.

Задача F. Теннисный турнир

В теннисном турнире участвуют 2^n игроков ($1 \leq n \leq 18$). Каждый теннисист имеет индивидуальный числовой рейтинг. Чем больше рейтинг, тем выше мастерство игрока. Известно, что рейтинги всех игроков разные.

Турнир проходит по олимпийской системе с выбываниями. После каждого тура проигравшие покидают турнир. Таким образом, n -й тур является финалом.

Организаторы формируют начальный список участников. В первом туре первый игрок списка играет со вторым, третий с четвертым и т. д. В результате список сокращается вдвое. Во втором туре встречаются победители первой и второй пар, третьей и четвертой и т. д. Далее все продолжается аналогично. Таким образом, начальный список и результаты матчей полностью определяют пары участников последующих встреч.

Игроки получили стартовые номера в порядке убывания их рейтинга. Сильнейший игрок имеет номер 1, второй по силе — номер 2 и т. д. Конечно, никому не хочется, чтобы сильные игроки встречались на ранних этапах. Более того, по традиции проведения турниров фавориты играют сначала со слабейшими игроками. Поэтому в первом туре запланированы матчи игроков с номерами 1 и 2^n , 2 и $2^n - 1$, 3 и $2^n - 2$ и т. д. Решено подобный принцип распространить и на последующие туры.

Организаторы рассуждают так. Допустим, что в матчах всегда будут побеждать игроки с более высоким рейтингом. Тогда во второй тур выйдут игроки с номерами 1,

2, ..., 2^{n-1} . Желательно, чтобы далее встретились игроки с номерами 1 и 2^{n-1} , 2 и $2^{n-1} - 1$, ..., 2^{n-2} и $2^{n-2} + 1$. Тогда в третий тур выйдут лучшие игроки с номерами 1, 2, ..., 2^{n-2} . При подобном течении турнира в каждый следующий тур будут попадать лучшие по рейтингу игроки предыдущего тура. Интрига сохранится до конца турнира, а в финале окажутся первые два номера рейтинга.

Помогите организаторам турнира сформировать начальный список игроков в соответствии с описанными пожеланиями.

Входные данные.

В единственной строке находится значение n .

Выходные данные.

В единственной строке вывести через пробел 2^n номеров игроков, составляющих начальный список.

В случае нескольких вариантов вывести любое допустимое решение.

Примеры.

№ п/п	Ввод	Вывод
1	1	1 2
2	2	1 4 2 3

Описание алгоритма.

Разделим 2^n игроков на две группы по их номерам следующим образом:

- 1 $\Rightarrow$ группа 1;
- 2, 3 $\Rightarrow$ группа 2;
- 4, 5 $\Rightarrow$ группа 1;
-
- $2^n - 2$, $2^n - 1$ $\Rightarrow$ группа 2;
- 2^n $\Rightarrow$ группа 1.

В обеих группах будет по 2^{n-1} игроков.

Далее каждую группу снова разделим пополам по тому же принципу. Получим 4 подгруппы по 2^{n-2} игроков.

Продолжим дальнейшее деление до получения пар.

Пусть, например, имеется $2^4 = 16$ игроков. При делении на восьмерки получим:

1, 4, 5, 8, 9, 12, 13, 16	2, 3, 6, 7, 10, 11, 14, 15
---------------------------	----------------------------

После деления на четверки будет:

1, 8, 9, 16	4, 5, 12, 13
2, 7, 10, 15	3, 6, 11, 14

Далее делим четверки на пары. Получаем:

1, 16	8, 9
4, 13	5, 12
2, 15	7, 10
3, 14	6, 11

Итак, начальная расстановка игроков с разбивкой на пары в порядке их получения составит: (1, 16), (8, 9), (4, 13), (5, 12), (2, 15), (7, 10), (3, 14), (6, 11). Если всегда будут выигрывать сильнейшие игроки, то во втором туре встретятся: (1, 8), (4 5), (2 7), (3 6), в третьем туре: (1, 4), (2, 3), а в финале 1 и 2. Все по справедливости!

Задача Г. Жизнь на Марсе

При высадке на Марс было основано n поселений. Каждое из них равномерно расширялось во все стороны на l марсианских миль в месяц. Постепенно поселения начали сливаться друг с другом, получая общее название. Какое минимальное время с момента высадки потребуется для того, чтобы на Марсе осталось не более k поселений?

Входные данные.

В первой строке задаются через пробел три целых положительных значения: начальное количество поселений n ($2 \leq n \leq 1000$), число k ($1 \leq k \leq 10$, $k < n$) и скорость роста поселений l ($1 \leq l \leq 100$).

В следующих n строках содержатся через пробел целые координаты поселений x_i , y_i ($-1000 \leq x_i$, $y_i \leq 1000$) в марсианских милях.

Выходные данные.

Вывести с точностью до трех знаков после запятой (точки) минимальное время в месяцах с момента высадки, необходимое для того, чтобы в результате слияния осталось не более k поселений.

Пример.

№ п/п	Ввод	Вывод
1	3 2 1 -1 1 2 1 2 5	1.500
2	3 1 1 -1 1 2 1 2 5	2.000

Описание алгоритма.

Будем рассматривать граф, соответствующий марсианским поселениям. Если нет совпадающих по координатам поселений, то сначала граф состоит только из n вершин. Два поселения соприкоснутся в середине соединяющего их отрезка через время $t = d/2l$, где d — расстояние между поселениями. Назовем это событие встречей. Будем встречу поселений отмечать ребром графа.

Поселения, расположенные ближе друг от друга, будут встречаться раньше. Нельзя считать, что при каждой встрече число поселений уменьшается на 1. Во-первых, при равных ребрах некоторые встречи будут происходить одновременно. Во-вторых, появление нового ребра может не приводить к уменьшению числа поселений, если обе вершины на концах этого ребра уже связаны через более короткие ребра.

Задача сводится к поиску такого минимального множества самых коротких расстояний между точками, что граф из n вершин и соответствующих ребер будет состоять из k или менее компонент связности.

Отсортируем по возрастанию $n \cdot (n-1)/2$ расстояний между точками. Бинарным поиском определим минимальное время, через которое граф будет содержать не более k компонент связности. В качестве начального отрезка выберем $[0, t_{\max}]$, где $t_{\max} = d_{\max}/2l$, а $d_{\max}$ — максимальное расстояние между точками. Для текущего значения времени t построим ребра только между теми поселениями, которые встретятся не более чем за время t .

Для определения числа компонент связности графа достаточно обойти его в глубину (алгоритм DFS) [4, 6, 8, 9, 11]. Трудоемкость этого метода $O(V+E)$, где V — число вершин, а E — число ребер. В нашем случае получим трудоемкость $O(n^2)$.

Сложность как сортировки, так и нахождения минимального времени с применением бинарного поиска — $O(n^2 \log n)$, что определяет асимптотику общего объема вычислений.

Задача Н. Игра в ферзи

Вася и Петя долго разбирались в задаче о расстановке на шахматной доске восьми ферзей, не бьющих друг друга. В подобных расстановках каждые два ферзя должны находиться на различных горизонталях, вертикалях и диагоналях. Потом они придумали следующую игру на эту тему.

Имеется шахматная доска размером $n \times n$ ($3 \leq n \leq 12$). Горизонтали и вертикали доски пронумерованы от 1 до n . Начинающий игрок первым ходом ставит ферзя на первую горизонталь. Второй игрок вторым ходом ставит следующего ферзя на вторую горизонталь так, чтобы он не бил первого ферзя. Далее первый игрок третьим ходом располагает очередного ферзя на третьей горизонтали так, чтобы он не бил предыдущих ферзей. И так далее. Игрок выигрывает, если ему удастся поставить n -го ферзя либо у противника не оказывается следующего хода. Каждый из игроков собираются выиграть как можно быстрее. Если выигрыш недостижим, то проигрывающий для сохранения достоинства должен проиграть через максимально возможное число ходов.

Вася начинает игру и своим первым ходом ставит ферзя в позицию l первой горизонтали. Требуется определить победителя игры, а также количество и последовательность ходов. Каждый ход состоит в указании номера позиции очередного ферзя.

Входные данные.

В единственной строке содержатся значения n и l ($1 \leq l \leq n$).

Выходные данные.

В первой строке выводится 1, если выигрывает Вася, либо 2 в случае выигрыша Пети.

Во второй строке выводится количество ходов партии.

В третьей строке выдается последовательность ходов в виде номеров позиций.

Если возможно несколько различных последовательностей длины k , то указывается та из них, которая меньше в лексикографическом (алфавитном) порядке. Иными словами, меньшей считается та последовательность, для которой меньше k -значное число из номеров позиций в системе счисления с основанием $n + 1$.

Примеры.

№ п/п	Ввод	Вывод
1	3 2	1 1 2
2	3 3	2 2 3 1

Описание алгоритма.

Поиском с возвратом [9, 11] найдем все конечные позиции игры. Их не слишком много. Так, для $n = 12$ даже при произвольном первом ходе конечных позиций менее 315 тысяч, т. е. при заданном первом ходе их приблизительно на порядок меньше. Значит, можно построить полное дерево игры. Корнем дерева будет позиция после первого заданного в условии хода Васи. В этой позиции свой первый ход должен сделать Петя.

Присвоим оценки конечным позициям, соответствующим листьям дерева. Пусть позиция достигнута после k ходов. При нечетном k дадим позиции оценку 1. Это выигрышная для Васи позиция. При четном k оценим позицию значением -1 .

Будем поднимать оценки вверх по дереву в соответствии с принципом минимакса [9]. В вершине, соответствующей ходу Васи, оценка определяется как максимум оценок в сыновьях. Аналогично в вершине, соответствующей ходу Пети, оценка составляет минимум оценок в сыновьях. Полученная оценка a в корне дерева определяет результат игры: выигрыш Васи при $a = 1$ или выигрыш Пети при $a = -1$.

Как модифицировать оценки в соответствии с условиями задачи? Достаточно все оценки 1 изменить на $100 - k$, а оценки -1 на $-100 + k$. Такие оценки позволяют каждому игроку в первую очередь стремиться к выигрышу, а во вторую очередь — к минимальному количеству ходов при выигрыше и к максимальному — при проигрыше.

Пусть снова в корне получена оценка a . Положительное значение a означает выигрыш Васи после $100 - a$ ходов. Отрицательное значение a соответствует выигрышу Пети через $100 + a$ ходов.

Нужно сохранять вместе с оценкой соответствующий ход игрока. Если при построении дерева и расчете рассматривать позиции в строках по возрастанию номеров, то последовательность номеров будет удовлетворять условию задачи.

Результаты и выводы

По сравнению с 2019 годом [2] в олимпиаде текущего года были предложены более трудные задачи. Тем не менее победители из Самары добились абсолютного результата, решив все восемь задач и набрав 800 баллов из 800 возможных. Призерами стали школьники из Татарстана и Удмуртии, набравшие 570 и 565 баллов.

Только победителям удалось полностью справиться с задачами D, E и H, закончив на последних минутах решение задачи D. Наиболее доступными ожидаемо оказались задачи B и F, за которые полные 100 баллов получили 12 и 13 команд соответственно. Каждая команда сдала на 100 баллов хотя бы одну задачу.

В ряде команд участники программировали почти все время либо совместно, либо независимо друг от друга, однако призеры показали более правильное командное взаимодействие. При подготовке к подобным олимпиадам стоит большее внимание уделять тактике командной работы и приемам решения олимпиадных задач, используя опыт студенческих команд [1, 7]. Вместе с тем школьникам в отличие от студентов приходится искать «золотую середину» между полным решением и более простым частичным, позволяющим набрать какие-либо баллы.

Не все участники правильно оценивали вычислительную сложность своих алгоритмов. Наряду с привычной недооценкой сложности были случаи, когда перспективные подходы отвергались.

При составлении олимпиадных задач принималось в расчет, что некоторые программы могут быть написаны на языках Pascal или Python и работать медленнее, поэтому размерность исходных данных была несколько уменьшена. Однако 13 команд из 14 использовали язык C++ и только одна команда программировала на Python. Олимпиада еще раз подтвердила, что для успешного выступления необходимо хорошее владение возможностями библиотеки шаблонов STL языка C++.

Регионы по-разному подошли к формированию команд. В некоторых командах, возможно, в ущерб сиюминутным интересам, были школьники VII—VIII классов.

Список использованных источников

1. Акишев И. Р. Об опыте участия в командных соревнованиях по программированию формата ACM ICPC // Информатика. 2008. № 19. С. 20–28. http://is.ifmo.ru/works/_akishev.pdf
2. Галочкин В. И. Задачи Интеллектуальной олимпиады Приволжского федерального округа среди школьников //

Информатика в школе. 2019. № 8. С. 56–62. DOI: 10.32517/2221-1993-2019-18-8-56-72

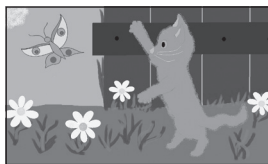
3. Кирюхин В. М., Цветкова М. С. Информатика. Программы внеурочной деятельности учащихся по подготовке к Всероссийской олимпиаде школьников: 5–11 классы. М.: БИНОМ. Лаборатория знаний, 2014. 224 с.
4. Кормен Т., Лайзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ: 3-е изд.: пер. с англ. М.: Вильямс, 2013. 1328 с.
5. Меньшиков Ф. В. Олимпиадные задачи по программированию. СПб.: Питер, 2003. 315 с.
6. Окулов С. М. Программирование в алгоритмах. М.: БИНОМ. Лаборатория знаний, 2006. 383 с.
7. Оршанский С. А. Орешении олимпиадных задач по программированию формата ACM ICPC // Информатика. 2006. № 1. <http://inf.1september.ru/article.php?ID=200600106>
8. Порублев И. Н., Ставровский А. Б. Алгоритмы и программы. Решение олимпиадных задач. М.: Вильямс, 2007. 480 с.
9. Рейнгольд Э. Н., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980. 476 с.
10. Сайт системы ejudge. https://ejudge.ru/wiki/index.php/Система_ejudge
11. Скиена С. С., Ревилла М. А. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. М.: КУДИЦ-ОБРАЗ, 2005. 416 с.

КОНКУРСЫ

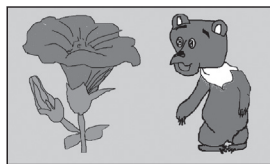
В оформлении обложки данного номера журнала «Информатика в школе» использованы работы следующих авторов — победителей конкурса цифровых изображений и фотографий:



Р. Н. Гаджиева,
учитель информатики средней
общеобразовательной
школы № 34, г. Махачкала,
Республика Дагестан



Ольга Метелёва,
ученица гимназии № 4
им. А. С. Пушкина,
г. Йошкар-Ола,
Республика Марий Эл



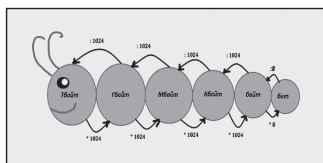
Ксения Носонова,
ученица Наро-Фоминской
средней общеобразовательной
школы № 1,
Московская область



Мария Казакова,
ученица средней
общеобразовательной
школы № 20, г. Черногорск,
Республика Хакасия



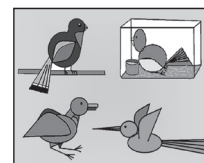
Анна Брагина,
ученица гимназии № 1,
г. Новосибирск



Тимофей Утилеев,
ученик Павловской гимназии,
Московская область



Алексей Панько,
ученик Оренбургского президент-
ского кадетского училища



Алиса Мадьярова,
ученица Павловской гимназии,
Московская область



Татьяна Чагаева,
учащаяся объединения «Компьютерная
графика и анимация» Центра
художественных ремесел, ученица средней
общеобразовательной школы № 36,
г. Дзержинск, Нижегородская область



Кристина Новикова,
учащаяся станции юных техников
имени Героя Социалистического Труда
Б. Г. Никитина, ученица средней
общеобразовательной школы № 10,
г. Воткинск, Удмуртская Республика



Анастасия Михайлова,
студентка Мордовского
государственного педагогического
института имени М. Е. Евсевьева,
г. Саранск, Республика Мордовия