Муниципальный этап Всероссийской олимпиады школьников по информатике

в 2016 – 2017 учебном году

Разборы решений и идеи тестов

Муниципальный этап Всероссийской олимпиады школьников по информатике в 2016 – 2017 учебном году

9 класс

Время выполнения задач — 4 часа

Ограничение по времени — 2 секунды на тест

Ограничение по памяти — 64 мегабайта

9.1. «Арифметический пример». Три целых числа a, b и c нужно так подставить вместо звёздочек в выражение $* \cdot (* + *)$, чтобы получилось максимально возможное значение. Напишите программу, которая по заданным величинам находит это значение.

Формат входа: В единственной строке через пробел заданы три целых числа a, b, c, по модулю не превосходящих 30000.

Формат выхода: Выведите единственное целое число — наибольший возможный результат выражения.

Пример

Вход:	*Выход:*
`-10 -2 5`	`10`

9.2. «Перенос слова». Как известно, одним из мест возможного переноса слова является место, где идут подряд две одинаковых буквы, если только при этом на предыдущей или последующей строке не остаётся одного символа от слова. Петя Торопыжкин, разрабатывая программу лингвистического анализа, хочет иметь процедуру, которая подсчитывает по заданному слову количество таких мест в заданной строке. Помогите ему, напишите соответствующую программу.

Формат входа: В единственной строке задано слово, состоящее из заглавных букв латиницы. Длина слова не менее 1 и не более 255 символов.

Формат выхода: Выведите единственное целое число — количество мест в строке, где подряд идут два одинаковых символа.

Пример 1

Вход:	*Выход:*
`BBAAAC`	`2`

Пример 2

Вход:	*Выход:*
`ABC`	`0`

Примечание: В первом примере нельзя перенести слово между двумя буквами `B`.

9.3. «Совокупный НОД». На уроках математики Петя Торопыжкин прошёл тему делимости целых чисел. В частности, он изучил процедуру нахождения НОД — наибольшего общего делителя — двух натуральных чисел. Теперь ему

хочется создать процедуру нахождения НОД трёх натуральных чисел. Помогите ему, напишите соответствующую программу.

Формат входа: В единственной входной строке через пробел указаны три натуральных числа k_1, k_2, k_3 ($1 \leqslant k_1, k_2, k_3 \leqslant 10^9$).

Формат выхода: Выведите единственное натуральное число НОД(k_1, k_2, k_3).

Пример

Вход:	*Выход:*
`100 35 80`	`5`

9.4. «Собираем букет». Петя Торопыжкин собирается на день рождения своей подруги Маши и хочет подарить ей букет, состоящий не более, чем из k цветов. В его распоряжении имеется n цветов. Зная предпочтения своей подруги, Петя может сказать какое удовольствие принесёт ей тот или иной цветок (каждая оценка — целое число, возможно, отрицательное: некоторые цветы Маша не любит). Помогите Пете составить букет — найдите наибольшее возможное суммарное Машино удовольствие от букета. В букете должен быть хотя бы один цветок!

Формат входа: В первой строке через пробел задано два целых числа: n — количество цветов у Пети — и k — максимальное количество цветов в букете ($1 \leqslant n \leqslant 10^5$, $1 \leqslant k \leqslant n$). Во второй строке через пробел задано n целых чисел, по модулю не превосходящих 10^4 — оценок машиного удовольствия от каждого из имеющихся цветков.

Формат выхода: Выведите единственное целое число — наибольшее удовольствие, которое может получить Маша от собранного Петей букета.

Пример 1

Вход:	*Выход:*
`3 3` `5 -10 6`	`11`

Пример 2

Вход:	*Выход:*
`3 3` `-100 -6 -2`	`-2`

9.5. «Добраться до базы». На поверхности планеты проложена прямая дорога, ведущая к базе. На этой дороге введена одномерная система координат. Вследствие землетрясения некоторые участки дороги обвалились, так что от дороги остались только два отрезка, $[0, l]$ и $[l + p, l + p + r]$. На правом отрезке расположена база, а на левый в точке с целой координатой x_0 и целой горизонтальной скоростью v_0 приземлился вездеход, которому нужно добраться до базы. Точнее, просто оказаться на правом плато и остановиться — там его уже подберут. Движение вездехода осуществляется следующим образом. В начале очередного секундного промежутка времени он должен проделать ровно одно из следующих действий:

- мгновенно изменить свою скорость на $+1$, подав команду `A`;
- мгновенно изменить свою скорость на -1, подав команду `D`;

- сохранить свою скорость, подав команду `K`;
- подпрыгнуть с сохранением скорости, подав команду `J`, и передвигаться следующую секунду, не касаясь поверхности (при преодолении провала между плато нужно подпрыгнуть; в остальные моменты движения можно подпрыгивать по желанию — это не запрещается, но и не помогает).

Затем робот передвигается на расстояние, равное модулю полученной скорости, в сторону, соответствующую знаку полученной скорости («+» — передвигается вправо, «−» — передвигается влево). Затем, если цель не достигнута, наступает новый цикл движения. Однако если после передвижения вездеход оказывается вне обоих плато (то есть его координата меньше 0, больше $l + p + r$ или лежит в интервале $(l, l+p)$), то он проваливается в пропасть и разбивается. Также он проваливается в пропасть, если при движении он проходит центральный провал — интервал $(l, l + p)$, не находясь в состоянии прыжка. Однако возможности вездехода подобраны таким образом, что он может успешно перепрыгнуть пропасть и остановиться на правом плато.

Ваша задача — написать программу, которая по заданным параметрам местности и условиях приземления вездехода составит последовательность команд, ведущую к успешному завершению движения вездехода.

Формат входа: В единственной строке через пробел заданы пять целых чисел: l, p, r (параметры плато: $1 \leqslant l, r \leqslant 3000$, $1 \leqslant p \leqslant 80$), x_0, v_0 (параметры приземления: $0 \leqslant x_0 \leqslant l$, $|v_0| \leqslant 100$). Все величины таковы, что вездеход может выполнить свою задачу.

Формат выхода: Выведите единственную строку, не включающую других символов, кроме `A`, `D`, `K`, `J`, являющуюся какой-либо программой для вездехода, приводящей его к успеху — полной остановке на правом плато без провалов в пропасть в промежуточные моменты. Гарантируется, что имеется разумное решение, состоящее не более чем из 10000 команд, и результат не должен включать бóльшее количество команд.

Пример 1		**Пример 2**		**Пример 3**	
Вход:	*Выход:*	*Вход:*	*Выход:*	*Вход:*	*Выход:*
`3 2 4 0 0`	`AAJDD`	`3 2 4 3 -1`	`KKKAAAJDD`	`3 2 4 0 4`	`DJDDD`

Примечание: Во втором примере есть и другие программы, приводящие к успеху.

Муниципальный этап Всероссийской олимпиады школьников по информатике в 2016 – 2017 учебном году
9 класс. Разбор решений и идеи тестов

9.1. «Арифметический пример». *Три целых числа a, b и c нужно так подставить вместо звёздочек в выражение $* \cdot (* + *)$, чтобы получилось максимально возможное значение. Напишите программу, которая по заданным величинам находит это значение.*

Задача, по мнению программного комитета, является утешительной. Возможно как лобовое решение, так или иначе рассматривающее все три принципиально различных варианта подстановки, так и «интеллектуальное» решение, анализирующее знаки и модули чисел a, b, c.

Идеи тестов:

1. $a, b, c > 0$.
2. $a, b, c < 0$.
3. $a, b, c = 0$.
4. $a < 0$, $b, c > 0$.
5. $b < 0$, $a, c > 0$.
6. $c < 0$, $a, b > 0$.
7. $a > 0$, $b, c < 0$.
8. $b > 0$, $a, c < 0$.
9. $c > 0$, $a, b < 0$.
10. $a = 0$, $b, c > 0$.
11. $a = 0$, $b > 0$, $c < 0$.
12. $a = 0$, $b, c < 0$.
13. $b = 0$, $a, c > 0$.
14. $b = 0$, $a > 0$, $c < 0$.
15. $b = 0$, $a, c < 0$.
16. $c = 0$, $a, b > 0$.
17. $c = 0$, $a > 0$, $b < 0$.
18. $c = 0$, $a, b < 0$.
19. $a = b = 0$, $c > 0$
20. $a = b = 0$, $c < 0$

9.2. «Перенос слова». *Как известно, одним из мест возможного переноса слова является место, где идут подряд две одинаковых буквы, если только при этом на предыдущей или последующей строке не остаётся одного символа от слова. Петя Торопыжкин, разрабатывая программу лингвистического анализа, хочет иметь процедуру, которая подсчитывает по заданному слову количество*

таких мест в заданной строке. Помогите ему, напишите соответствующую программу.

Данная задача является простой, по мнению программного комитета. Подразумевается лобовое решение: проход по строке с сравнением предыдущего и последующего символов; при этом поиск начинается со второго символа и заканчивается предпоследним, чтобы исключить подсчет совпадений, дающих одиночный символ после переноса.

Идеи тестов:

1. Односимвольная строка
2. Двухсимвольная строка из разных символов
3. Двухсимвольная строка из одинаковых символов
4. Тест без повторений, длина строки меньше максимума.
5. Тест без повторений, длина строки максимальна.
6. Тест с однократными повторениями, не совпадают первый и второй символы и предпоследний и последний, длина строки меньше максимума.
7. Тест с однократными повторениями, не совпадают первый и второй символы и предпоследний и последний, длина строки максимальна.
8. Тест с однократными повторениями, совпадают первый и второй символы, не совпадают предпоследний и последний, длина строки меньше максимума.
9. Тест с однократными повторениями, совпадают первый и второй символы, не совпадают предпоследний и последний, длина строки максимальна.
10. Тест с однократными повторениями, не совпадают первый и второй символы, совпадают предпоследний и последний, длина строки меньше максимума.
11. Тест с однократными повторениями, не совпадают первый и второй символы, совпадают предпоследний и последний, длина строки максимальна.
12. Тест с многократными повторениями, не совпадают первый и второй символы и предпоследний и последний, длина строки меньше максимума.
13. Тест с многократными повторениями, не совпадают первый и второй символы и предпоследний и последний, длина строки максимальна.
14. Тест с многократными повторениями, совпадают первый и второй символы, не совпадают предпоследний и последний, длина строки меньше максимума.
15. Тест с многократными повторениями, совпадают первый и второй символы, не совпадают предпоследний и последний, длина строки максимальна.
16. Тест с многократными повторениями, не совпадают первый и второй символы, совпадают предпоследний и последний, длина строки меньше максимума.
17. Тест с многократными повторениями, не совпадают первый и второй символы, совпадают предпоследний и последний, длина строки максимальна.

18. Все символы в строке одинаковы, длина строки меньше максимума.
19. Все символы в строке одинаковы, длина строки максимальна.

20–25. Случайные тесты.

9.3. «Совокупный НОД». *На уроках математики Петя Торопыжкин прошёл тему делимости целых чисел. В частности, он изучил процедуру нахождения НОД — наибольшего общего делителя — двух натуральных чисел. Теперь ему хочется создать процедуру нахождения НОД трёх натуральных чисел. Помогите ему, напишите соответствующую программу.*

Задача подразумевает техническое решение, включающее написание процедуры, реализующей классический алгоритм Евклида нахождения НОД двух целых чисел. Впрочем, в некоторых языках (например, Python) может присутствовать соответствующая функция. Поэтому с точки зрения программного комитета, данная задача имет сложность ниже средней.

Идеи тестов:

1–5. Случайные тесты с НОД $= 1$.

6–20. Случайные тесты с НОД > 1.

9.4. «Собираем букет». *Петя Торопыжкин собирается на день рождения своей подруги Маши и хочет подарить ей букет, состоящий не более, чем из k цветов. В его распоряжении имеется n цветов. Зная предпочтения своей подруги, Петя может сказать какое удовольствие принесёт ей тот или иной цветок (каждая оценка — целое число, возможно, отрицательное: некоторые цветы Маша не любит). Помогите Пете составить букет — найдите наибольшее возможное суммарное Машино удовольствие от букета. В букете должен быть хотя бы один цветок!*

Данная задача требует определенных действий по своей формализации, а также привлекает не вполне тривиальную идею (как в теоретическом, так и в практическом плане) для своего решения. Вследствие этого программный комитет считает, что сложность этой задачи несколько выше средней.

Формализованная постановка выглядит следующим образом. Имеется набор целых чисел, требуется выбрать из них не менее одного и не более k, чтобы сумма чисел из выбранного набора была наибольшей. Ясно, что если в наборе есть неотрицательные числа, ответ — их сумма, поскольку при их суммировании результат неубывает с увеличением количества слагаемых. Если же в наборе все числа отрицательные, то нужно взять одно наибольшее, поскольку при суммировании отрицательных чисел сумма убывает с увеличением количества слагаемых.

Наиболее оптимальная идея решения выглядит следующим образом: отсортируем числа по убыванию. Если наибольшее число отрицательное, возвращаем его

в качестве результата. Иначе суммируем числа с начала массива, пока не наберётся k слагаемых или пока очередной элемент массива не станет отрицательным. Для реализации этой идей для получения полного балла требуется написать хороший алгоритм сортировки (или использовать библиотечную функцию), неоптимальные, квадратичные сортировки (типа пузырьковой) не пройдут по времени на тестах, где количество элементов более 14-15 тысяч. Участники, аккуратно реализовавшие идею решения, но с неоптимальной сортировкой, получат около 75% полного балла.

Идеи тестов:

1. Одно положительное число, $k = 1$.
2. Один ноль, $k = 1$.
3. Одно отрицательное число, $k = 1$.
4. Два положительных числа, $k = 1$.
5. Положительное число и ноль, $k = 1$.
6. Положительное и отрицательное число, $k = 1$.
7. Два ноля, $k = 1$.
8. Ноль и отрицательное число, $k = 1$.
9. Два отрицательных числа, $k = 1$.
10. Два положительных числа, $k = 2$.
11. Положительное число и ноль, $k = 2$.
12. Положительное и отрицательное число, $k = 2$.
13. Два ноля, $k = 2$.
14. Ноль и отрицательное число, $k = 2$.
15. Два отрицательных числа, $k = 2$.
16. Случайный тест, $n \approx 5000$, все числа неотрицательные, $1 < k < n$.
17. Случайный тест, $n \approx 5000$, есть отрицательные числа, k меньше количества неотрицательных чисел.
18. Случайный тест, $n \approx 5000$, есть отрицательные числа, k больше количества неотрицательных чисел.
19. Случайный тест, $n \approx 5000$, все числа отрицательные, $1 < k < n$.
20. Случайный тест, $n \approx 25000$, все числа неотрицательные, $1 < k < n$.
21. Случайный тест, $n \approx 25000$, есть отрицательные числа, k меньше количества неотрицательных чисел.
22. Случайный тест, $n \approx 25000$, есть отрицательные числа, k больше количества неотрицательных чисел.
23. Случайный тест, $n \approx 25000$, все числа отрицательные, $1 < k < n$.
24. Максимальный тест, $n = 10^5$, есть отрицательные числа, k меньше количества неотрицательных чисел.
25. Максимальный тест, $n = 10^5$, есть отрицательные числа, k больше количества неотрицательных чисел.

9.5. «Добраться до базы». *На поверхности планеты проложена прямая дорога, ведущая к базе. На этой дороге введена одномерная система координат. Вследствие землетрясения некоторые участки дороги обвалились, так что от дороги остались только два отрезка,* $[0, l]$ *и* $[l + p, l + p + r]$*. На правом отрезке расположена база, а на левый в точке с целой координатой* x_0 *и целой горизонтальной скоростью* v_0 *приземлился вездеход, которому нужно добраться до базы. Точнее, просто оказаться на правом плато и остановиться — там его уже подберут. Движение вездехода осуществляется следующим образом. В начале очередного секундного промежутка времени он должен проделать ровно одно из следующих действий:*

- *мгновенно изменить свою скорость на* $+1$*, подав команду* `A`*;*
- *мгновенно изменить свою скорость на* -1*, подав команду* `D`*;*
- *сохранить свою скорость, подав команду* `K`*;*
- *подпрыгнуть с сохранением скорости, подав команду* `J`*, и передвигаться следующую секунду, не касаясь поверхности (при преодолении провала между плато нужно подпрыгнуть; в остальные моменты движения можно подпрыгивать по желанию — это не запрещается, но и не помогает).*

Затем робот передвигается на расстояние, равное модулю полученной скорости, в сторону, соответствующую знаку полученной скорости («+» — передвигается вправо, «−» — передвигается влево). Затем, если цель не достигнута, наступает новый цикл движения. Однако если после передвижения вездеход оказывается вне обоих плато (то есть его координата меньше 0, больше $l+p+r$ *или лежит в интервале* $(l, l + p)$*), то он проваливается в пропасть и разбивается. Также он проваливается в пропасть, если при движении он проходит центральный провал — интервал* $(l, l+p)$*, не находясь в состоянии прыжка. Однако возможности вездехода подобраны таким образом, что он может успешно перепрыгнуть пропасть и остановиться на правом плато.*

Ваша задача — написать программу, которая по заданным параметрам местности и условиях приземления вездехода составит последовательность команд, ведущую к успешному завершению движения вездехода.

По мнению программного, комитета данная задача является весьма сложной, хотя допускает относительно простое частичное решение, которое для некоторых вариантов входных данных существенным образом использует тот факт, что решение обязательно существует.

Рассмотрим вспомогательный факт. Если в какой-то момент вездеход стоит неподвижно, то чтобы набрать скорость v ему потребуется v команд `A` (или `D`). За это время он пройдет расстояние

$$1 + 2 + 3 + \cdots + v = \frac{v(v + 1)}{2}$$

(по формуле суммы арифметической прогрессии). Наоборот, если вездеход обладает скоростью v, то при максимальном торможении до полной остановки он

пройдет такой путь.

Назовём *минимальным прыжком* прыжок с края левого плато на край правого плато. Понятно, что для минимального прыжка необходимо, чтобы вездеход находился в точке с координатой l и обладал скоростью p. Из условия существования решения следует, что после прыжка и приземеления на правом плато вездеход успеет остановиться. То есть, что длина r правого плато не меньше $\frac{p(p+1)}{2}$. Если длина правого плато больше, чем эта величина, то могут быть совершены неминимальные прыжки — прыжки с большей скоростью и, как следствие, не с края левого плато или не на край правого.

Опишем теперь случаи, на которые разумно разделить входные параметры.

1) В начальный момент вездеход покоится, начальная его скорость равна нулю. Из условия существования решения в этом случае следует, что длина l левого плато не меньше, чем $\frac{p(p+1)}{2}$, то есть позволяет набрать скорость для минимального прыжка.

Соответственно, прямолинейное решение в этом случае состоит в том, чтобы ускориться на одну единицу влево или вправо к точке с координатой $l - \frac{p(p+1)}{2}$, то есть к точке, откуда может быть набрана скорость для минимального прыжка и остановиться там. После чего начать ускоряться направо, сделать прыжок, оказавшись на краю левого плато, а после прыжка начать тормозить до полной остановки.

2) Вначале скорость вездехода отрицательна. Из условия существования решения в этом случае следует, что длина левого плато и положение вездехода на нём таковы, что ему хватит длины плато, чтобы остановится. А раз так, то длина плато достаточна для того, чтоб разогнаться для минимального прыжка.

Соответственно, прямолинейное решение в этом случае состоит в том, чтобы остановиться, а затем действовать как в случае 1).

3) В начальный момент скорость вездехода положительна, но его положение и скорость таковы, что оставшейся длины левого плато хватит для его остановки, то есть $n - x_0 \geqslant \frac{v_0(v_0+1)}{2}$. Если при этом длина левого плато достаточна для набора скорости минимального прыжка, то есть если $l \geqslant \frac{p(p+1)}{2}$, то тогда прямолинейное решение состоит в том, чтобы остановиться, а потом действовать в соответствии с пунктом 1).

4) Наконец, пусть скорость вездехода в начальный момент положительна, но оставшейся части плато недостаточно, чтобы остановиться, или длины левого плато недостаточно, чтобы набрать скорость минмиального прыжка. В этом случае надо набирать скорость минимального (или неминимального) прыжка без остановки.

Именно эта подзадача является весьма сложной: проблема поиска управления переводящего систему из одного заданного положения в другое непроста, задачами такого рода занимается раздел математики под названием теория управления.

Однако в нашем случае из-за дискретности задачи — дискретности времени и конечности набора управлений — задачу можно решить перебором. Может быть использована следующая рекурсивная процедура, которая принимает текущее состояние вездехода — координату x и скорость v, перебирает управления, вызывает саму себя с новыми возможными положениями и возвращается строку управлений, ведущих к успеху, или пустую строку, если из данного состояния вездеход не может достигнуть цели.

1. Если вездеход уже перепрыгнул центральный провал, то есть если $x \geqslant l + p$, то

   1.1. Пытаемся тормозить и смотрим, то будет дальше: за время торможения вездеход пройдёт путь $\frac{(v-1)v}{2}$, соответственно, торможение будет успешным, если $x+\frac{(v-1)v}{2} \leqslant l+p+r$. Если торможение будет успешным, выдаём строку, состоящую из символов `D` в количестве v штук, иначе выдаём пустую строку.

2. Иначе мы ещё находимся на левом плато.
3. Если мы можем перепрыгнуть центральный провал, то есть если $x+v \geqslant l+p$, то вызываем себя с параметрами $x' = x + v$, $v' = v$; если результат — пустая строка, то неудача, возвращаем пустую строку; иначе — успех, возвращаем полученную строку, дописав в начало `J`.
4. Прямо сейчас перепрыгнуть не можем. Проверим, сможем ли набрать скорость минимального прыжка, то есть если $v < p$, то должно быть выполнено условие $x + \frac{((v+1)+p)(p-v)}{2} \leqslant l$. Если это условие нарушено, то перепрыгнуть не сможем, возвращаем пустую строку.
5. Иначе потенциально перепрыгнуть можем. Перебираем возможные управления:

   5.1. Пробуем сохранить скорость: проверяем, что не переедем центральный провал и не попадём в него, $x + v \leqslant l$, и вызываем себя с параметрами $x' = x+v$, $v' = v$; если результат — непустая строка, то успех, возвращаем полученную строку, дописав в начало `K`.

   5.2. Пробуем ускориться: проверяем, что не переедем центральный провал и не попадём в него, $x + (v + 1) \leqslant l$, и вызываем себя с параметрами $x' = x + (v + 1)$, $v' = v + 1$; если результат — непустая строка, то успех, возвращаем полученную строку, дописав в начало `A`.

   5.3. Если скорость не равна 1, $v \neq 1$, пробуем затормозить: проверяем, что не переедем центральный провал и не попадём в него, $x + (v - 1) \leqslant l$, и вызываем себя с параметрами $x' = x+(v-1)$, $v' = v-1$; если результат — непустая строка, то успех, возвращаем полученную строку, дописав в начало `D`.

6. Ни одно изменение скорости не привело к успеху — неудача, возвращаем пустую строку.

Запуск данной процедуры из начального состояния $x = x_0$, $v = v_0$ даст решение, которое по условию существует.

Оценим сложность данной процедуры. Количество шагов по времени, за которое при нулевой начальной скорости и постоянном ускорении пересекается левое плато при максимальной его длине, находится из уравнения $\frac{k(k+1)}{2} = 3000$ и примерно равно 80. На каждом шаге мы перебираем три возможности — ускорение, сохранение скорости, замедление, что дает общее число операций около $3^{80} \approx 10^{47}$, что, конечно же, является неподъёмным перебором. Однако, при данных, когда левое плато пересекается за не слишком большое число шагов (16–18 шагов; $3^{16} \approx 10^8$, $3^{18} \approx 10^9$), этот перебор может быть осуществлён за разумное время. Тем не менее, программа, аккуратно реализующая описанную процедуру получит полный балл, поскольку происходит разумное отсечение вариантов и чтобы заставить эту процедуру работать на полном или почти полном переборе нужны особенно искусные тесты.

Потенциальный недостаток указанного переборного решения заключается в том, что одни и те же состояния анализируются многократно. Например, если из текущего состояния (x, v) мы сначала замедлимся, потом ускоримся, а потом сохраним скорость, то перейдём в состояние $(x+(v-1)+v+v = x+3v-1, v)$. В такое же состояние мы перейдем, если сначала сохраним скорость, потом замедлимся, а затем ускоримся: $(x+v+(v-1)+v = x+3v-1, v)$. Соответственно, если после первого попадания в такое состояние успеха не удалось достичь, то состояние надо пометить как бесперспективное. Тогда при втором попадании мы прочтём эту метку и во второй раз не станем проводить перебор управлений из этого положения. Тем самым возникает идея мемоизации (запоминания) результатов, и мы приходим к принципу динамического программирования.

Другой взгляд на мемоизацию результатов может дать формализация в рамках теории графов. Рассмотрим граф, вершинами которого являются возможные состояния вездехода (x, v). Здесь $x \in [0, l] \cup [l + p, l + p + r]$, $|v| \leqslant 200$. Задача состоит в том, чтобы найти путь от начальной вершины (x_0, v_0) до одной из целевых вершин, которые составляют набор $\{(x, 0) : x \in [l+p, l+p+r]\}$. При этом гарантируется, что путь существует. Если добавить идею помечивания посещённых вершин, то процедура, аналогичная приведённой выше, фактически будет реализовывать обход графа посредством алгоритма поиска в глубину.

При этом общая сложность алгоритма с мемоизацией будет пропорциональна общему количеству возможных состояний вездехода, то есть

$$(l + r) \cdot 2v_{\max} = 2 \cdot 3000 \cdot 200 = 1\,200\,000,$$

что значительно меньше, чем сложность переборного алгоритма без мемоизации.

Соответственно, пусть у нас имеется двумерный массив меток состояний вездехода `МЕТКА[0..`$l+p+r$`, -100..+100]`; первый индекс соответвует координате вездехода, второй — его скорости (для удобства индексирования введём и ячейки,

соответствующие координатам среднего провала). Каждая ячейка хранит логическое значений. При этом размер массива составит чуть больше мегабайта, что укладывается в ограничения по памяти. Значение `false` в некоторой ячейке присваивается в том случае, когда в состоянии, соответствующем этой ячейке, мы уже побывали, `true` — ещё не побывали. То есть, если в ячейке записано `false`, то переходить в это состояние не нужно. В начале положим `METKA[i,j] := false` при `i` $= l+1, \ldots, l+p-1$, `j` $= -100, \ldots, +100$, то есть в состояния, находящиеся в среднем провале, переходить нельзя.

Переборная процедура, так же возвращающая строку программы, выглядит следующим образом:

1. Процедура вызвана с состоянием (x, v); при этом считаем, что состояние допустимое по координате и по скорости.
2. Метим текущее состояние как обработанное: `METKA[`x`,` v`] := false`;
3. Если вездеход находится на правом плато и скорость равна 1, то мы можем замедлиться и остановиться, то есть достигнуть цели. Если $l+p \leqslant x \leqslant l+p+r$ и $v == 1$, то вернуть строку `D`.
4. Если вездеход находится на правом плато, то нет нужды пытаться разгоняться, следует только пытаться тормозить: если $l+p \leqslant x \leqslant l+p+r$, то

   4.1. если $x + (v-1) \leqslant l+p+r$ и `METKA[`$x + (v-1)$`, `$v-1$`] == true`, то вызвать себя с состоянием $(x + (v-1), v-1)$. (Не проверяем, что скорость положительна, так как на правое плато мы можем попасть только с положительной скоростью.) Если результат пустая строка (неуспех), вернуть пустую строку. Иначе приписать к полученной строке спереди `D` и вернуть результат.

   4.2. Если условие из предыдущего пункта не выполнено, то есть мы или выкатываемся за правый край правого плато, или попадаем в состояние, ранее проанализированное, то неуспех, возвращаем пустую строку.

5. Иначе мы ещё находимся на левом плато.
6. Если мы можем перепрыгнуть центральный провал, то есть если $x+v \geqslant l+p$ и `METKA[`$x+v$`, `v`] == true`, то вызываем себя с состоянием $(x+v, v)$; если результат — пустая строка, то неудача, возвращаем пустую строку; иначе — успех, возвращаем полученную строку, дописав в начало `J`.
7. Прямо сейчас перепрыгнуть не можем. Перебираем возможные управления:

   7.1. Пробуем ускориться: проверяем, что скорость не превзойдет крайнего значения, что не переедем центральный провал и что не попадём в него, а также что не съедем с плато влево, $v < v_{\max}$ и $0 \leqslant x + (v+1) \leqslant l$ и `METKA[`$x + (v+1)$`, `$v+1$`] == true`, и вызываем себя с состоянием $(x+(v+1), v+1)$; если результат — непустая строка, то успех, возвращаем полученную строку, дописав в начало `A`.

7.2. Иначе пробуем сохранить скорость: проверяем, что не переедем центральный провал и не попадём в него, а также что не съедем с плато влево, $0 \leqslant x + v \leqslant l$ и `МЕТКА[`$x + v$`, `v`] == true`, и вызываем себя с состоянием $(x + v, v)$; если результат — непустая строка, то успех, возвращаем полученную строку, дописав в начало `K`.

7.3. Иначе пробуем затормозить: проверяем, что скорость не превзойдет крайнего значения, что не переедем центральный провал и что не попадём в него, а также что не съедем с плато влево, $v > -v_{\max}$ и $0 \leqslant x+(v-1) \leqslant l$ и `МЕТКА[`$x + (v - 1)$`, `$v - 1$`] == true`, и вызываем себя с состоянием $(x + (v - 1), v - 1)$; если результат — непустая строка, то успех, возвращаем полученную строку, дописав в начало `D`.

8. Ни одно из возможных управлений не привело к успеху; значит, из данного состояния успех невозможен. Возвращаем пустую строку.

Запуск данной процедуры из начального состояния (x_0, v_0) даст решение, которое по условию существует.

Видно, что в данных условиях, в принципе, существует разумное решение длины не более 3200: даже если вездеход стоит с нулевой скоростью на правом краю левого плато, которое имеет максимальную длину, и выходит на траекторию минимального прыжка самым неоптимальным образом — получив отрицательную скорость 1 и сохраняя её до выхода в точку старта, то здесь он потратит около 3000 шагов, а дальнейший энергичный разгон, прыжок и энергичное торможение потребуют не более 200 шагов.

Предлагаемый набор тестов таков, что доля тестов с коротким плато, которые могут быть решены при помощи процедуры, основанной на физических идеях, составляет 60%. Впрочем, некоторые тесты с длинным левым плато тоже решаются наивной физической процедурой при некоторых её реализациях, так что участник при помощи такой процеду может заработать около двух третей полного балла.

Идеи тестов:

1. Провал без внутренности: $l = p = r = 1$, вездеход покоится на левом краю левого плато: $x_0 = 0$, $v_0 = 0$.
2. Провал без внутренности: $l = p = r = 1$, вездеход разогнан вправо на левом краю левого плато: $x_0 = 0$, $v_0 = 1$.
3. Провал без внутренности: $l = p = r = 1$, вездеход сильно разогнан вправо на левом краю левого плато: $x_0 = 0$, $v_0 = 2$, неминимальный прыжок.
4. Провал без внутренности: $l = p = r = 1$, вездеход разогнан влево на левом краю левого плато: $x_0 = 0$, $v_0 = -1$.
5. Провал без внутренности: $l = p = r = 1$, вездеход покоится на правом краю левого плато: $x_0 = 1$, $v_0 = 0$.

6. Провал без внутренности: $l = p = r = 1$, вездеход разогнан вправо на правом краю левого плато: $x_0 = 1$, $v_0 = 1$.
7. Провал без внутренности: $l = p = r = 1$, вездеход разогнан влево на правом краю левого плато: $x_0 = 1$, $v_0 = -1$.
8. Провал без внутренности: $l = p = r = 1$, вездеход сильно разогнан влево на правом краю левого плато: $x_0 = 1$, $v_0 = -2$.
9. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход покоится на левом краю левого плато: $x_0 = 0$, $v_0 = 0$.
10. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход покоится на правом краю левого плато: $x_0 = 55$, $v_0 = 0$.
11. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход небыстро едет вправо в середине плато: $x_0 = 25$, $v_0 = 3$, нужно остановиться и выйти на траекторию минимального прыжка.
12. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход небыстро едет влево в середине плато: $x_0 = 25$, $v_0 = -3$, нужно остановиться и выйти на траекторию минимального прыжка.
13. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато на траектории разгона на минимальный прыжок: $x_0 = 28$, $v_0 = 7$.
14. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато, чуть опережая траекторию разгона на минимальный прыжок: $x_0 = 28$, $v_0 = 8$.
15. Длина левого плато в точности равна дистанции набора скорости минимального прыжка: $l = 55$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато быстрее, чем требуется для разгона на минимальный прыжок, но успевает притормозить: $x_0 = 28$, $v_0 = 9$.
16. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход покоится на левом краю левого плато, но длина правого плато позволяет затормозить только после минимального прыжка: $x_0 = 0$, $v_0 = 0$.
17. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход покоится на правом краю левого плато: $x_0 = 75$, $v_0 = 0$.
18. Длина левого плато больше дистанции набора скорости минимального

прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход небыстро едет вправо в левой части плато: $x_0 = 10$, $v_0 = 3$.

19. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход небыстро едет вправо в левой части плато: $x_0 = 15$, $v_0 = 3$.
20. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход небыстро едет вправо в середине плато: $x_0 = 45$, $v_0 = 3$, нужно остановиться и выйти на траекторию минимального прыжка.
21. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход небыстро едет влево в середине плато: $x_0 = 35$, $v_0 = -3$, нужно остановиться и выйти на траекторию минимального прыжка.
22. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато на траектории разгона на минимальный прыжок: $x_0 = 48$, $v_0 = 7$.
23. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато, чуть опережая траекторию разгона на минимальный прыжок: $x_0 = 48$, $v_0 = 8$.
24. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато быстрее, чем требуется для разгона на минимальный прыжок, но успевает притормозить: $x_0 = 48$, $v_0 = 9$.
25. Длина левого плато больше дистанции набора скорости минимального прыжка, но длина правого плато позволяет затормозить только после минимального прыжка: $l = 75$, $p = 10$, $r = 45$; вездеход быстро едет вправо в середине плато быстрее, чем требуется для разгона на минимальный прыжок, но есть возможность погасить скорость: $x_0 = 28$, $v_0 = 11$.
26. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход покоится на левом краю левого плато, длина правого плато позволяет затормозить и после

неминимального прыжка: $x_0 = 0$, $v_0 = 0$.

27. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход покоится на правом краю левого плато: $x_0 = 75$, $v_0 = 0$.
28. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход небыстро едет вправо в левой части плато: $x_0 = 10$, $v_0 = 3$.
29. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход небыстро едет вправо в левой части плато: $x_0 = 15$, $v_0 = 3$.
30. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход небыстро едет вправо в середине плато: $x_0 = 35$, $v_0 = 3$.
31. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход небыстро едет влево в середине плато: $x_0 = 35$, $v_0 = -3$.
32. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход быстро едет вправо в середине плато на траектории разгона на минимальный прыжок: $x_0 = 48$, $v_0 = 7$.
33. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход быстро едет вправо в середине плато, чуть опережая траекторию разгона на минимальный прыжок: $x_0 = 48$, $v_0 = 8$.
34. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход быстро едет вправо в середине плато быстрее, чем требуется для разгона на минимальный прыжок: $x_0 = 48$, $v_0 = 9$.
35. Длина левого плато больше дистанции набора скорости минимального прыжка, длина правого плато позволяет затормозить и после неминимального прыжка: $l = 75$, $p = 10$, $r = 85$; вездеход быстро едет вправо в середине плато быстрее, чем требуется для разгона на минимальный прыжок: $x_0 = 48$, $v_0 = 11$.

36–62. Те же идеи, что в тестах 9–35, только левое плато длинное — при макми-

альном разгоне из положения покоя проходится больше, чем за 20 шагов (в минимальном варианте $l = 465$, $p = 30$, $r = 435$).

63–90. Случайные тесты с коротким левым плато.

91–100. Случайные тесты с длинным левым плато, включая максимальные тесты.