Задача A. Спуск по лестнице

Idea: Илья Молочко
Development: Илья Молочко
Tutorial: Денис Ким

Expected difficulty: trivial
Success rate: 98.2 % (221/225 + 2 заморожено)

First to solve: 0:00:31 (BSUIR 2 : begunki)
Shortest solution: 32 байта (Belarusian NTU: Wrong Answer on Test 1)
Shortest jury solution 49 байт

Keywords: простая арифметика

Между этажами n и $n+1$ ровно x ступенек, значит, и между любой парой соседних этажей ровно x ступенек. Спуск с этажа $n+1$ на этаж 1 состоит из n пролётов между соседними этажами, поэтому ответ равен $n \cdot x$.

Сложность — $\mathcal{O}(1)$.

Задача B. Метро

Idea: Илья Молочко
Development: Илья Молочко
Tutorial: Денис Ким

Expected difficulty: trivial
Success rate: 67.9 % (186/274 + 20 заморожено)

First to solve: 0:03:44 (Belarusian SU: static_cast<AC>)
Shortest solution: 133 байта (Belarusian NTU: Skill Issue)
Shortest jury solution 338 байт

Keywords: простая арифметика

Все перегоны между станциями i и $i+1$ при $1 \leqslant i < n$ соединяют числа разной чётности, поэтому каждый из них стоит 1. Перегон между станциями n и 1 соединяет числа одной чётности тогда и только тогда, когда n нечётно: в этом случае он стоит 2, иначе 1.

Пусть $d = |s - f|$. Одна из двух дуг кольца идёт от меньшего номера к большему и не проходит через перегон между n и 1; на ней ровно d перегонов стоимости 1, так что она стоит d. Обратная дуга содержит $n - d$ перегонов и обязательно использует перегон между n и 1. Её стоимость равна $n - d$, если n чётно, и $n - d + 1$, если n нечётно. Ответ — минимум стоимостей двух дуг.

Сложность — $\mathcal{O}(t)$ ($\mathcal{O}(1)$ на тестовый случай).

Задача C. Вывески

Idea: Ефим Климашевский
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: trivial
Success rate: 82.6 % (194/235 + 5 заморожено)

First to solve: 0:01:38 (Belarusian SU: Team spilis)
Shortest solution: 171 байт (BSUIR: AgRE$ty)
Shortest jury solution 148 байт

Keywords: строки

Подстрока длины 3 является палиндромом тогда и только тогда, когда её первый и третий символы совпадают: средний символ может быть любым. Поэтому достаточно проверить все окна $s_i s_{i+1} s_{i+2}$: если хотя бы для одного i выполнено $s_i = s_{i+2}$, ответ `YES`, иначе `NO`.

Сложность — $\mathcal{O}(\sum |s|)$.

Задача D. Логотип

Idea: Алексей Толстиков
Development: Андрей Лукашевич
Tutorial: Денис Ким

Expected difficulty: very easy
Success rate: 68.0 % (157/231 + 14 заморожено)

First to solve: 0:08:48 (Grodno SU #1: Bl9ShK4)
Shortest solution: 331 байт (Vitebsk SU: 1/3_Braincell)
Shortest jury solution 732 байта

Keywords: реализация, конструктив

Чтобы найти минимальное m, нужно округлить вверх квадратный корень из n.

Заполнить поле $m \times m$ можно, например, таким способом. Заполним квадрат $(m-1) \times (m-1)$ в левом верхнем углу символами '#' и разместим оставшиеся $k = n - (m-1)^2$ символов '#' в L-образной рамке из $2m-1$ клеток (правый столбец и нижняя строка). Обойдём рамку как путь сверху вниз по правому столбцу, затем справа налево по нижней строке (клетку в правом нижнем углу не повторяем) и заполним первые k клеток пути. Пустые клетки образуют суффикс этого пути и поэтому связны; при $n = m^2$ заполняется вся таблица.

Сложность — $\mathcal{O}(n)$.

Задача E. Магический массив

Idea: Илья Молочко
Development: Илья Молочко
Tutorial: Денис Ким

Expected difficulty: very easy
Success rate: 27.5 % (112/408 + 87 заморожено)

First to solve: 0:08:38 (Belarusian SU: Team spilis)
Shortest solution: 292 байта (Brest STU: Borisevich's theorem)
Shortest jury solution 478 байт

Keywords: инвариант, сортировки

Соседние элементы можно менять местами тогда и только тогда, когда они разной чётности. При обмене нечётное число «проскакивает» через чётное. Инвариант этой операции — относительный порядок нечётных элементов и относительный порядок чётных. Наоборот, любую перестановку, сохраняющую инвариант, можно получить такими обменами.

Массив можно сделать неубывающим тогда и только тогда, когда нечётные элементы уже идут в неубывающем порядке и чётные — тоже: любое нарушение этого порядка — часть инварианта. Это проверяется одним проходом по массиву.

Сложность — $\mathcal{O}\left(\sum n\right)$.

Задача F. Фильтр для сообщений

Idea: Иван Лукьянов
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: very easy
Success rate: 37.6 % (59/157 + 45 заморожено)

First to solve: 0:20:36 (Belarusian SU: Team spilis)
Shortest solution: 989 байт (Belarusian SU: 128 burbalka fans)
Shortest jury solution 1137 байт

Keywords: строки, реализация, перебор

Смех задаётся гласной v, согласной c и видом: начинается с гласной или с согласной. Гласных 5, согласных 21, видов 2, итого $5 \cdot 21 \cdot 2$ кандидатов. Для фиксированных букв и вида максимальную чередующуюся подпоследовательность можно построить жадно слева направо: каждый раз берётся

самое левое вхождение нужной буквы правее уже взятых. Откладывать вхождение невыгодно — более левая позиция оставляет больше свободы справа. Если полученная длина чётна, последний символ отбрасывается, чтобы смех начинался и заканчивался одной и той же буквой. Сложность такого решения — $\mathcal{O}(|\Sigma|^2 \cdot |s|)$, где $|\Sigma|$ — размер используемого алфавита.

Можно (но не обязательно) решать и быстрее. Строка кандидата длины ℓ однозначно восстанавливается по первой букве, второй букве и ℓ. Поэтому среди кандидатов положительной длины выбирается наибольшая длина, а при равенстве длин — меньшая первая буква и, если $\ell > 1$, меньшая вторая. Все пары можно вести одновременно одним проходом по s: при чтении символа обновляются длины тех пар, в которые он входит. Сложность — $\mathcal{O}(|\Sigma| \cdot |s|)$.

Задача G. Кратное и делитель

Idea: Иван Лукьянов
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: easy
Success rate: 29.9 % (81/271 + 91 заморожено)

First to solve: 0:08:06 (BSUIR: Prime limit)
Shortest solution: 493 байта (Grodno SU #12: RMStar)
Shortest jury solution 363 байта

Keywords: теория чисел, сортировки

Отсортируем массив: $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_{2n}$. Число, делящее x, не превосходит x, а число, кратное x, не меньше x, поэтому $a_n \leqslant x \leqslant a_{n+1}$. Следовательно, n наименьших чисел должны делить x, а x должен делить n наибольших. Таким образом, $L = \text{НОК}(a_1, \ldots, a_n)$ делит ответ, а ответ делит $G = \text{НОД}(a_{n+1}, \ldots, a_{2n})$. Подходящее x существует тогда и только тогда, когда L делит G; в этом случае годится любое x с условием $L \mid x \mid G$.

В качестве ответа можно взять G: он всегда делит верхнюю половину, а если какое-то число нижней половины не делит G, то не подойдёт и никакой делитель G. Подойдёт и L, но при его вычислении нужно следить за переполнением: если НОК становится больше 10^9, ответа нет.

Сложность — $\mathcal{O}(n \log n)$.

Задача H. Угадай букву

Idea: Иван Лукьянов
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: easy
Success rate: 32.7 % (86/263 + 61 заморожено)

First to solve: 0:19:17 (Belarusian SU: simply lovely)
Shortest solution: 316 байт (Gomel SU: C++_++))
Shortest jury solution 428 байт

Keywords: двоичный поиск

Будем хранить множество букв, которые ещё могут быть загаданы. Сначала это все 26 маленьких латинских букв.

Каждый запрос состоит из текущего множества кандидатов. Если их число нечётно, допишем одну букву, которая кандидатом уже не является, чтобы длина стала чётной. Загаданная буква есть в запросе, поэтому жюри обязано удалить ровно одно её вхождение. Значит, в ответе этой буквы уже нет, и всех вернувшихся кандидатов можно вычеркнуть. Так множество сокращается примерно вдвое.

Пока кандидатов больше одного, повторяем запрос. Так потребуется $\lceil \log_2 |\Sigma| \rceil$ запросов, где $|\Sigma|$ — размер алфавита. Когда осталась одна буква, выводим её.

Сложность самой программы — $\mathcal{O}(|\Sigma|)$.

Задача I. Лайки

Idea: Алексей Толстиков
Development: Андрей Лукашевич
Tutorial: Денис Ким

Expected difficulty: easy-medium
Success rate: 37.8 % (59/156 + 68 заморожено)

First to solve: 0:22:46 (Belarusian SU: It's all decided. Accept it and relax!)
Shortest solution: 614 байт (Belarusian SU: static_cast<AC>)
Shortest jury solution 439 байт

Keywords: два указателя, префиксные суммы

Положим $d_0 = 0$ и $s_0 = 0$. Для каждого $j = 1, \ldots, n$ на отрезке дней $d_{j-1}+1, \ldots, d_j$ суммарно ровно $s_j - s_{j-1}$ лайков, и их можно произвольно распределить по дням этого отрезка неотрицательными целыми.

Чтобы максимизировать сумму на одном окне длины T, все лайки отрезка выгодно поместить в один его день. Тогда окно забирает целиком массу каждого пересекающегося с ним отрезка и ничего из остальных. Ответ равен максимуму $s_j - s_{i-1}$ по парам $i \leqslant j$, для которых существует окно длины T, пересекающее и i-й, и j-й отрезки (тогда оно пересечёт и все промежуточные).

Такое окно существует тогда и только тогда, когда $i = j$ либо $T \geqslant d_{j-1} - d_i + 2$: достаточно накрыть день d_i и день $d_{j-1} + 1$, и этот отрезок длины $d_{j-1} - d_i + 2$ всегда можно расширить до окна длины T внутри отрезка дней $1, \ldots, d_n$. Префиксные суммы не убывают, поэтому для фиксированного j достаточно минимального подходящего i: чем левее левый конец, тем больше разность $s_j - s_{i-1}$. При увеличении j это минимальное i не убывает. Все лучшие пары можно найти методом двух указателей: указатели i и j соответствуют концам рассматриваемого набора отрезков и одновременно движутся слева направо — j перебирает правый конец, а i сдвигается вправо до наименьшего индекса, при котором пара (i, j) допускает окно длины T.

Сложность — $\mathcal{O}(n)$.

Можно решить задачу иначе: перебрать все начала окон через разностный массив. Отрезок дней $d_{j-1}+1, \ldots, d_j$ пересекает окно с началом x тогда и только тогда, когда $d_{j-1}-T+2 \leqslant x \leqslant d_j$. Добавим массу отрезка на этот диапазон, обрезанный по $[1, d_n-T+1]$, и возьмём максимум префиксных сумм. Это решение работает за $\mathcal{O}(n + d_n)$ времени, расход памяти составляет $\mathcal{O}(n + d_n)$.

Задача J. Антиматерия

Idea: Андрей Лукашевич
Development: Андрей Лукашевич
Tutorial: Денис Ким

Expected difficulty: easy-medium
Success rate: 51.4 % (19/37 + 30 заморожено)

First to solve: 1:26:42 (Belarusian SU: simply lovely)
Shortest solution: 716 байт (Belarusian SU: Rizz godz)
Shortest jury solution 538 байт

Keywords: биты, префиксные суммы

Для $A = 0, 1, 2, 3$ оптимальны отрезки $[0, 0]$, $[0, 1]$, $[2, 2]$, $[0, 2]$ соответственно. Далее $A \geqslant 4$. Пусть B — старший бит числа A ($B \geqslant 4$).

Пусть $s(n) = 0 \oplus 1 \oplus \ldots \oplus n$ при $n \geqslant 0$ и $s(-1) = 0$. Будем искать полуинтервал $(L, R]$ ($L \geqslant -1$), то есть числа от $L + 1$ до R. Тогда

$$(L + 1) \oplus \ldots \oplus R = s(R) \oplus s(L),$$

а в конце прибавим 1 к L. Очевидно, что $(A - 1, A]$ удовлетворяет условию задачи, но мы попробуем улучшить этот ответ.

Поскольку $(4k) \oplus (4k + 1) \oplus (4k + 2) \oplus (4k + 3) = 0$, значение $s(n)$ определяется остатком n по модулю 4:

$n \bmod 4$	$s(n)$	$s(n) \bmod 4$
0	n	0
1	1	1
2	$n+1$	3
3	0	0

Заметим, что старшие биты n входят в $s(n)$ тогда и только тогда, когда n чётно.

Рассмотрим остатки L и R по модулю 4 и выпишем, чьи старшие биты берутся в итоговое значение (сверху в клетке) и каковы два младших бита ответа (снизу в клетке).

$L \setminus R$	0	1	2	3
0	L, R 0	L 1	L, R 3	L 0
1	R 1	— 0	R 2	— 1
2	L, R 3	L 2	L, R 0	L 3
3	R 0	— 1	R 3	— 0

Наилучший для нас вариант — это тот, при котором старшие биты берутся сразу из L и из R (это возможно только при $A \equiv 0 \pmod 4$ и при $A \equiv 3 \pmod 4$). Тогда (если не считать два младших бита) мы можем положить $R = B$, а остальные старшие биты A положим в L.

Если старшие биты берутся только из L или только из R, то старшие биты R не могут быть вместе меньше, чем старшие биты A. Это лучшее, чего мы можем достичь, если остаток A от деления на 4 — 1 или 2.

Если старшие биты не берутся ни из L, ни из R, то мы можем получить только число, не превосходящее 3, и нам это не подходит.

Более формально, учитывая данные из таблицы выше, мы можем получить, что оптимальный ответ для разных остатков A выглядит так:

- $A \equiv 0 \pmod 4$: $R = B$, $L = A \oplus B$.
- $A \equiv 1 \pmod 4$: $R = A - 1$, $L = 1$.
- $A \equiv 2 \pmod 4$: $R = A$, $L = A - 1$ (тривиальное решение).
- $A \equiv 3 \pmod 4$: $R = B$, $L = (A \oplus B \oplus 3) + 2$.

В конце нужно не забыть прибавить 1 к L.

Сложность — $\mathcal{O}(t)$ ($\mathcal{O}(1)$ на тестовый случай).

Задача K. Картинная галерея

Idea: Андрей Лукашевич
Development: Андрей Лукашевич
Tutorial: Денис Ким

Expected difficulty: medium
Success rate: 59.1 % (26/44 + 17 заморожено)

First to solve: 1:10:32 (Belarusian SU: simply lovely)
Shortest solution: 1523 байта (Belarusian SU: Dance, horse)
Shortest jury solution 1738 байт

Keywords: структуры данных, сет

После обрезки картина (i, j) имеет размер $r_i \times c_j$, где $1 \leqslant r_i \leqslant h_{i,j}$ и $1 \leqslant c_j \leqslant w_{i,j}$ для всех i, j. Суммарная площадь обрезков равна $S - \left(\sum_i r_i\right)\left(\sum_j c_j\right)$, где $S = \sum_{i,j} h_{i,j} w_{i,j}$. Все r_i и c_j положительны, а ограничения на ряды и столбцы независимы, поэтому произведение сумм максимально, если взять наибольшие допустимые значения: $r_i = \min_j h_{i,j}$ и $c_j = \min_i w_{i,j}$.

Достаточно поддерживать сумму площадей S, сумму минимумов высот по рядам и сумму минимумов ширин по столбцам. При замене в клетке (x, y) меняются только ряд x и столбец y. Чтобы обновлять минимумы, в каждом ряду и каждом столбце храните упорядоченную структуру: в C++ — `std::multiset` (или `std::set` пар), в Java — `TreeMap` как мультимножество (или `TreeSet`). Одно и то же значение может встречаться несколько раз, поэтому удалять нужно ровно один экземпляр. Время работы составляет $\mathcal{O}((nm + k)\log(n + m))$, расход памяти составляет $\mathcal{O}(nm)$.

Вместо мультимножеств можно хранить массив высот в порядке рядов и массив ширин в порядке столбцов в деревьях отрезков: минимумы нужного ряда и столбца становятся запросами на непрерывных отрезках. Время работы составляет $\mathcal{O}(nm + k\log(nm))$, расход памяти составляет $\mathcal{O}(nm)$.

Во всех решениях не забывайте 64-битный тип данных.

Задача L. Турнир претендентов

Idea:	Ефим Климашевский
Development:	Ефим Климашевский, Денис Ким
Tutorial:	Денис Ким
Expected difficulty:	medium
Success rate:	50.0 % (2/4 + 7 заморожено)
First to solve:	1:56:29 (* Guest team: BSUIR2)
Shortest solution:	3505 байт (* Guest team: BSUIR2)
Shortest jury solution	2548 байт

Keywords: динамическое программирование по дереву, вероятности

Будем для каждого претендента i хранить вероятность p_i того, что он выиграл своё текущее поддерево (в листе $p_i = 1$). Когда встречаются левое и правое поддеревья, смесь действий победителя правого поддерева равна $\sum p_j \cdot s_j$ по претендентам справа, где s_j — его распределение камня, ножниц и бумаги. Вероятность выигрыша линейна по стратегии соперника, поэтому претенденту i слева достаточно сыграть против этой трёхмерной смеси: умножим p_i на вероятность победы s_i против неё. Симметрично обновим правую сторону. Из-за сбалансированности каждый претендент лежит в $\mathcal{O}(\log n)$ поддеревьях, и суммарное время — $\mathcal{O}(n \log n)$.

В корне смесь действий чемпиона — снова $\sum p_i \cdot s_i$. Если Светозар показывает камень, он побеждает с вероятностью $S + \frac{R}{2}$, где R и S — вероятности камня и ножниц у чемпиона; для ножниц и бумаги формулы получаются циклическим сдвигом.

Сложность — $\mathcal{O}(n \log n)$.

Может показаться естественным хранить в вершине только распределение действий победителя поддерева. Если у двух сыновей распределения A и B, левый побеждает с вероятностью

$$w = A_R B_S + A_S B_P + A_P B_R + \tfrac{1}{2}(A_R B_R + A_S B_S + A_P B_P),$$

и дальше можно заменить обоих на смесь $w \cdot A + (1 - w) \cdot B$. Так получается один «виртуальный» претендент со смешанной стратегией. Это **неверно**: победитель — конкретный человек, и в следующем матче разные претенденты из смеси имеют разные шансы против нового соперника. Общую смесь действий нельзя подставлять вместо личности.

Задача M. Бинарный массив и суммы

Idea:	Андрей Лукашевич
Development:	Андрей Лукашевич
Tutorial:	Денис Ким
Expected difficulty:	medium-hard
Success rate:	11.1 % (1/9 + 6 заморожено)
First to solve:	3:50:39 (Belarusian SU: Team spilis)
Shortest solution:	2473 байта (Belarusian SU: Team spilis)
Shortest jury solution	2932 байта

Keywords: графы, разностные массивы

Обозначим $p_0 = 0$ и $p_i = a_1 + \ldots + a_i$. Тогда $a_i = p_i - p_{i-1}$ и $0 \leqslant p_i - p_{i-1} \leqslant 1$, а запрос (l, r, s) равносилен равенству $p_r - p_{l-1} = s$.

Равенства на префиксных суммах поддерживаем системой непересекающихся множеств со смещениями: в одном классе значения p_i отличаются на известную константу. Если очередное равенство противоречит уже известным, ответа нет.

После этого остаются ограничения $0 \leqslant p_i - p_{i-1} \leqslant 1$. Если индексы i и $i-1$ лежат в одном классе, разность уже фиксирована, и достаточно проверить, что она равна 0 или 1. Если в разных, получаются разностные ограничения на значения корней вида $x_u - x_v \leqslant w$.

Такая система совместна тогда и только тогда, когда в графе с ребром $v \to u$ веса w нет отрицательного цикла. Цикл ищется алгоритмом Форда-Беллмана; расстояния дают какую-нибудь допустимую расстановку корней. По ней восстанавливаются p_i и биты a_i. Если цикл найден, массив восстановить нельзя.

Сложность — $\mathcal{O}(n^2 + m)$, так как после сжатия равенств в графе $\mathcal{O}(n)$ вершин и $\mathcal{O}(n)$ рёбер, сами же равенства обрабатываются за $\mathcal{O}(m)$.

Задача N. Гномы

Idea: Егор Дубовик
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: medium-hard
Success rate: 8.8 % (15/171 + 167 заморожено)

First to solve: 0:55:13 (Belarusian SU: simply lovely)
Shortest solution: 2156 байт (Belarusian SU: simply lovely)
Shortest jury solution 2469 байт

Keywords: структуры данных, дерево отрезков

Если значение x уже известно, число его вхождений на $[l, r]$ при присвоениях считается за $\mathcal{O}(\log n)$: деревом Фенвика по позициям (+1 в тех индексах, где стоит x) либо упорядоченным множеством этих позиций. В C++ подходит `ordered_set` из GNU pbds (`tree_order_statistics_node_update`), в Java — `TreeSet` или `TreeMap`. При присвоении a_p позицию p удаляют из множества старого значения и вставляют в множество нового. Расход памяти составляет $\mathcal{O}(n \log n)$.

Перебирать все различные числа на отрезке не нужно. Достаточно получить небольшое множество кандидатов на x, среди которых обязан лежать ответ, если он существует, и проверить каждый из них подсчётом частоты.

Кандидатов на x можно получить несколькими способами.

- **Голосование с одним счётчиком.** Идём по массиву и храним текущее значение y и целое c. Очередной элемент, равный y, увеличивает c; иначе c уменьшается, а при $c = 0$ в y записывается новый элемент и c становится равным 1. Если какое-то значение встречалось строго больше половины раз, оно останется в y: каждый чужой элемент уменьшает его счётчик не больше чем на единицу. Результат не зависит от порядка, поэтому ту же пару (y, c) хранят в узлах дерева отрезков и объединяют за $\mathcal{O}(1)$ (при совпадении значений счётчики складывают, иначе вычитают меньший из большего). Запрос на $[l, r]$ даёт одного кандидата на x; его частоту проверяют подсчётом. Для порога «строго больше половины» этого достаточно: такого значения не больше одного.

- **Голосование с двумя счётчиками.** Хранят не больше двух пар «значение, счётчик». Совпавшее значение увеличивает свой счётчик; новое занимает свободный слот, а если слотов нет, все счётчики уменьшают на единицу (нули отбрасывают). Значение, встречающееся больше чем в трети позиций, не может быть вытеснено полностью. Частота больше $1/2$ влечёт частоту больше $1/3$, поэтому ответ, если он есть, лежит среди не более чем двух кандидатов на x. Те же пары объединяют в дереве отрезков за $\mathcal{O}(1)$ и проверяют подсчётом частоты.

- **Медиана.** В отсортированном подотрезке значение с частотой больше $1/2$ занимает медианную позицию. k-ю порядковую статистику на отрезке при обновлениях в точке поддерживают деревом отрезков; найденное значение — кандидат на x, его проверяют подсчётом частоты. Такая структура работает за $\mathcal{O}((n+q)\log^2 n)$ и **не проходит** ограничение по времени.
- **Случайная выборка.** Значение с частотой больше $1/2$ на отрезке не больше одного. Достаточно взять несколько случайных позиций: оно появится в выборке с постоянной вероятностью. Все различные выбранные числа (их мало) — кандидаты на x, их проверяют тем же подсчётом. Такие решения могли не проходить все тесты в некоторых случаях.

Более прямолинейные решения работают за $\mathcal{O}((n+q)\log^2 n)$: в каждом узле дерева отрезков хранят словарь частот либо используют два вложенных дерева Фенвика. Это слишком медленно.

Сложность голосования и случайной выборки составляет $\mathcal{O}((n+q)\log n)$.