Задача A. Простое произведение

Из определения простого числа следует, что в ответ состоит из единицы и любого простого числа, не превосходящего n. Например, пара чисел 1 и 2 всегда будет ответом.

Задача B. Карточки с цифрами

Если у нас есть карточка с шестеркой, её нужно перевернуть и получить девятку. После этого нужно первой поставить карточку с большей цифрой, а второй карточку с меньшей цифрой.

Задача C. Самое большое число

Легко понять, что достаточно упорядочить три цифры по убыванию, а затем вывести их от наибольшего к наименьшему без разделительных пробелов.

Задача D. Бочки

Если слева стоят бочки с бензином, то мы возьмем их все, если $k > n_1$, иначе возьмем k бочек. То есть, если $c_1 = A$, то ответ $min(k, n_1)$.

Если слева стоят бочки с керосином, то мы не возьмем ни одной бочки с бензином, если $k \leqslant n_1$, иначе возьмем $k - n_1$ бочек бензина. То есть, если $c_1 = B$, то ответ $max(0, n_1 - k)$.

Задача E. Подготовка к контесту

Обозначим за $finish$ время, которое нужно Мише, чтобы решить все задачи. В первый час будет решено a задач, значит на остальное время останется $(n - a)$ задач. Таким образом, $finish = 60 + (n - a)b$ минут. Всего соревнование длится $60t$ минут.

Миша в точности успеет решить все задачи, если $finish = 60t$.

Миша успеет решить все задачи, и у него останется время, если $finish < 60t$, а останется у него как раз $60t - finish$ минут.

Если Миша не успеет решить все задачи ($finish > 60t$), то ему не хватит $finish - 60t$ минут. За время $60t$ Миша успеет решить a задач в первый час и $\frac{60(t-1)}{b}$ задач в последующие часы. Таким образом, у него останутся нерешенными $n - a - \frac{60(t-1)}{b}$ задач.

Задача F. Игра на повышение

Поскольку мы не можем убирать камни, попробуем уравнять все кучки до значения $max(a, b, c)$. За каждый ход мы прибавляем ровно два камня, поэтому четность суммы камней во всех кучках не изменится. Таки образом, если $max(a, b, c) - (a + b + c)$ — четное, то мы можем достигнуть во всех кучках значения $max(a, b, c)$ за $\frac{max(a,b,c)-(a+b+c)}{2}$ ходов.

Если $max(a, b, c)-(a+b+c)$ — нечетное, то мы можем сделать во всех кучках только $max(a, b, c)+1$ камень за $\frac{max(a+1,b+1,c+1)-(a+b+c)}{2}$ ходов.

Задача G. Хитмейкер

Если $n = 1$, то ответ одна единица.

Если n кратно трём, то ответ $n/3$ троек.

Если n имеет остаток один при делении на 3, то ответ две двойки и $(n - 4)/3$ тройки.

Если n имеет остаток два при делении на 3, то ответ одна двойка и $(n - 2)/3$ тройки.

Докажем, что это оптимальный ответ. Заметим следующие факты:

1. Среди звуков нет 1. Рассмотрим два слагаемых 1 и a можем заменить на одно $a + 1$, сумма не изменится, а произведение увеличится $a \cdot 1 < a + 1$.

2. Среди звуков нет чисел больших 4. Обозначим этот звук за x. Так как $x > 4$, следует, что $2x - 9 > 0$, то есть $x < 3x - 9 = 3 \cdot (x - 3)$. Значит можно заменить звук x на два звука 3 и $x - 3$, сохранив сумму и увеличив произведение.

3. Четверку можно заменить на две двойки, так как сумма и произведение не изменятся. Таким образом можно считать, что искомый набор звуков состоит только из двоек и троек.

4. Двоек не может быть больше двух, так как $2 + 2 + 2 = 3$, но $2 \cdot 2 \cdot 2 < 3 \cdot 3$.

Задача H. Тривиальность

Так как границы интервала — небольшие числа, то мы можем себе позволить пройтись по каждому числу с L до R, вычислить его тривиальность по формуле из условия и, если нужно, обновить максимум и минимум.

Чтобы вычислить тривильность, нужно перебрать все делители числа. Заметим, что все делители разбиваются на пары: если x является делителем числа A, то и $\frac{A}{x}$ является делителем числа A. При этом меньший делитель из пары не превосходит $\sqrt{A}$ (если x – младший делитель и $x \geqslant \sqrt{A}$, и $\frac{A}{x} > x \geqslant \sqrt{A}$, то $\frac{A}{x} \cdot x > \sqrt{A} \cdot \sqrt{A}$, чего быть не может).

Таким образом при переборе делителей числа A достаточно перебирать меньший делитель из пары до $\sqrt{A}$, находить парный ему делитель $\frac{A}{x}$ и аккуратно обрабатывать числа, являющиеся полным квадратом (если $\sqrt{A}$ это делитель, то у него нет пары, например, у числа 25 из условия).

Асимптотика программы $\mathcal{O}(R\sqrt{R})$.

Задача I. Трудно запомнить дни рождения

При считывании информации о друзья, будем поддерживать для каждого дня друга с максимальным значением c_i. Для этого можно использовать словарь или *map* в зависимости от языка, либо просто создать массив для каждого дня.

Каждый раз, когда вводится новый человек, мы сначала проверяем, не встречался ли человек с таким же днем рождения ранее. В этом случае мы запомним этого человека, иначе сравним значение c_i нового человека с имеющимся для этой даты максимумом. Обновим максимум если это требуется, и запомним имя нового человека.

Просмотрите все дни рождения и внесите в список имена друзей, которые ему больше всего нравятся. Отсортируйте список (в большинстве языков есть сортировка и сравнение в алфавитном порядке) и выведите.

Задача J. Богдан и подстроки

Назовем балансом строки разность между количеством вхождения в неё первого символа и другого символа. Тогда необходимо найти такую подстроку строки, в которой модуль баланса наибольший.

Посчитаем массив $bal[i]$ – баланс префикса данной строки s, заканчивающегося в позиции i. Тогда $bal[i] = bal[i-1] + 1$, если i-й символ равен 1-му, и $bal[i] = bal[i-1] - 1$ иначе.

Теперь как по этому массиву быстро вычислить баланс некоторой подстроки исходной строки s. Пусть нам нужно определить баланс подстроки с i по j. Если бы мы делали это проходом, мы бы начали с нуля и шли счетчиком как в предыдущем абзаце. Вместо этого, давайте просто посмотрим как изменится баланс, от позиции $i-1$ (предположили, будто бы на тот момент баланс 0), к позиции j. Эта величина равна $bal[j] - bal[i-1]$ и равна балансу подстроки $i..j$.

Таким образом нам нужно найти такие i и j, что $bal[j] - bal[i]$ – максимальное. Это соответствует подстроке $i{+}1..j$. Нетрудно заметить, что для этого достаточно взять максимальный и минимальный элементы в массиве bal (учитывая, что $bal[0] = 0$).

Получили линейное относительно размера входных данных решение (посчитать балансы префиксов, и найти в этом массиве максимум и минимум).

Задача K. Маленький читер

Сначала решим задачу за квадратичное время. Переберем задание, ответ на которое будем изменять. Пройдемся по всему массиву, и посчитаем НОД всех чисел, кроме изменяемого. Это значение и будет ответом для случая изменения этого элемента. Выберем то задание, которое дает максимальный ответ.

Такое решение не укладывается по времени. Давайте теперь будем определять НОД все чисел, кроме одного, быстрее, чем полным проходом по массиву. Предварительно посчитаем следующие две величины:

- $prefix[i] = gcd(a[1], a[2], \ldots, a[i]) = gcd(prefix[i-1], a[i])$
- $suffix[i] = gcd(a[i], \ldots, a[n-1], a[n]) = gcd(a[i], suffix[i+1])$

Тут gcd — НОД чисел. Теперь чтобы узнать НОД всех чисел, кроме элемента на позиции i, нужно посчитать $gcd(prefix[i-1], suffix[i+1])$. Такое решение работает за линейное время и укладывается в ограничение по времени.

Основная часть решения задачи на языке C++:

```
for (int i = 0; i < n; i++) {
    cin >> a[i + 1];
}
for (int i = 0; i < n; i++) {
    pref_gcd[i + 1] = gcd(a[i + 1], pref_gcd[i]);
    suff_gcd[n - i] = gcd(a[n - i], suff_gcd[n - i + 1]);
}
int ans = 1;
for (int i = 1; i <= n; i++) {
    ans = max(ans, gcd(pref_gcd[i - 1], suff_gcd[i + 1]));
}
```

Задача L. Стрельба по телевизорам

Было дано множество прямоугольников, расположенных вдоль оси ox и множество точек. Нужно было определить, сколько точек попадают в прямоугольники или лежит на границе.

Сначала выпишем в отдельный массив координаты всех правых верхних углов прямоугольников, и отсортируем точки запросов. Теперь будем идти двумя указателями. Для каждой очередной точки запроса, будем поддерживать указатель на прямоугольник под этой точкой, то есть на такой самый левый прямоугольник, что его правый верхний угол находится левее нашей точки (или имеет такую же x координату).

Для каждой точки будем увеличивать указатель на прямоугольник, пока правый верхний угол текущего прямоугольника левее, чем точка запроса. Теперь мы знаем, какой прямоугольник лежит под нашей точкой, и можем легко определить, попадаем ли мы на границу. Нужно учесть случай, когда точка запроса находится на той же x координате, что и правый верхний угол прямоугольника, и точка попадает на границу следующего прямоугольника. Если полоса прямоугольников закончилась, новые точки запросов можно не рассматривать.

Альтернативным решением было просто для каждой точки находить двоичным поиском прямоугольник внизу, тогда не требовалось сортировать точки изначально.

Задача M. Урок физкультуры

Построим граф учеников так, чтобы в одной компоненте связности ученики обязаны были надеть колпак одного цвета.

Сделаем это следующим образом:

- Сопоставим каждому из $n \times m$ учеников свою вершину в графе.
- Пусть a_1, a_2, …, a_k — номера вершин, сопоставленных с учениками одного класса. Проведем следующие ребра: (a_1, a_2), (a_2, a_3), …, (a_{k-1}, a_k). Тогда все вершины a_1, a_2, …, a_k окажутся в одной компоненте связности.
- Проведем ребра из ученика на позиции (i, j) в ученика на позиции $(i+1, j)$ для всех допустимых i, j. Таким образом ученики из одного столбца окажутся в одной компоненте связности.

В таком графе $\mathcal{O}(n \times m)$ вершин и ребер. Ответом на задачу будет являться количество компонент связности в таком графе, что можно вычислить с помощью обхода в глубину или обхода в ширину. https://ru.wikipedia.org/wiki/Компонента_связности_графа

Итоговое время работы $\mathcal{O}(n \times m)$

Задача N. Вова на даче

Давайте решим другую задачу — научимся вычислять функцию $f(x)$, равную количеству чисел от 1 до $x-1$, в битовой записи которых без лидирующих нулей ровно k нулевых битов. Тогда ответ на исходную задачу — это $f(R+1) - f(L)$.

Приступим к вычислению $f(x)$. Сначала рассмотрим все числа, битовая запись которых короче битовой записи x. Все они меньше x, следовательно, количество чисел длиной len с k нулями равно $\binom{len-1}{k}$ (так как первый бит обязан быть 1) (число сочетаний из $len-1$ по k).

Теперь рассмотрим числа такой же длины. Переберём первый слева бит, в котором x и наше потенциальное число различаются. Такие позиции соответствуют единичным битам в x, кроме самой старшей единицы, так как числа меньшей длины мы уже рассмотрели. Пусть битовая длина x равна len, мы рассмотрели t позиций префикса и стоим в позиции $t+1$, а также уже поставили p нулей, тогда на суффиксе длиной $len-t-1$ требуется расставить $k-p-1$ ноль. Это можно сделать $\binom{len-t-1}{k-p-1}$ способами.

Таким образом, требуется предподсчитать все цешки для $1, 2, \ldots, n$, где $n = \log_2 R$. Это можно легко сделать за квадратичное от n время, воспользовавшись формулой $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

Итоговое время работы — $\mathcal{O}(\log^2 R)$.

Задача O. Заснеженные дороги

Во-первых, заметим, что существует такое C, что для любого уровня снега меньше C граф остается связным, а для любого уровня снега не меньше C - становится несвязным. Это значение C можно найти, например, с помощью бинарного поиска по уровню снега и проверкой на связность любым алгоритмом обхода графа.

Во-вторых, при увеличении уровня снега количество опасных городов не уменьшается. Значит, для решения задачи достаточно сделать бинарный поиск по уровню снега, а затем найти количество опасных городов. По определению город является опасным тогда и только тогда, когда он является точкой сочленения в графе. Воспользуемся известным алгоритмом поиска точек сочленений в графе https://e-maxx.ru/algo/cutpoints и получим решение для все задачи с временной сложностью $\mathcal{O}((n+m)logC)$.