Задача A. Соревнование трёх

Для решения первой группы тестов ($n \leqslant 50$) достаточно перебрать для первых двух участников, сколько каждый из них занял первых мест и сколько — вторых. Тогда остальные значения (количество мест у каждого участника) считаются однозначно и, тем самым, мы получим количество очков победителя. Сложность алгоритма $O(n^4)$.

Для решения полной задачи нужно заметить, что практически всегда мы можем добиться оптимального результата, то есть победитель наберет $\lceil \frac{sum}{3} \rceil$, где sum — суммарное количество очков участников. Действительно, можно сначала раздать каждому из участников по $\lfloor \frac{n}{3} \rfloor$ первых, вторых и третьих мест. Далее у нас может остаться еще 1 или 2 этапа. Разность очков, полученную на этих двух этапах, можно исправить, меняя вторую позицию на третью у победителя (разница с предыдущим местом изменится на 2 очка). Таким образом, формула верна при $n \geqslant 4$. Несложно проверить, что эта же формула подходит и для случаев $n = 2$, $n = 3$.

Задача B. Ленивые голубцы

Для решения первой группы тестов ($n \leqslant 10^9$) можно промоделировать, сколько должен пробежать каждый голубец. Поскольку мы получим арифметическую прогрессию, то сумма ее элементов будет расти квадратично от длины. Таким образом, максимальный ответ не будет превосходить $O(\sqrt{n})$.

Для написания более быстрого решения можно воспользоваться алгоритмом бинарного поиска по ответу. Действительно, чем больше голубцов, тем больше расстояние суммарно они должны пробежать. При фиксированном числе голубцов c мы получим сумму арифметической прогрессии $s_c = \frac{c \cdot (1 + (1 + k \cdot (c-1)))}{2}$, и эта сумма должна быть не больше значения s из входных данных. Таким образом, оптимальное значение c можно найти за $O(ln(s))$ операций. При этом в последней подгруппе тестов можно получить переполнение 64-битного типа данных. Чтобы избежать этой ситуации, можно либо воспользоваться 128-битными (или более) типами данных, или заменить умножение делением (сделать замену выражения $a \cdot b \leqslant c$ на $a \leqslant \frac{c}{b}$).

Задача C. Правильные тройки

Для первой группы тестов достаточно перебрать все тройки чисел a, b, c и для каждой проверить, выполняется ли хотя бы одно из требуемых равенств.

Для второй группы можно перебрать пары чисел a, b. Тогда мы получим два возможных значения c, которые нужно проверить (выполняется ли условие $1 \leqslant c \leqslant n$). Тут также важно не забыть, что при равенстве значений нужно учитывать только одно из них.

Для увеличения скорости работы стоит перебирать значение c вместо пары значений a и b. При этом для каждого c количество решений уравнения $a + b = c$ будет ровно $(c - 1)$ (слагаемое a может принимать значения от 1 до $c - 1$). Для подсчета количества решений уравнения $a \cdot b = c$ необходимо найти количество делителей числа c. В третьей подзадаче это можно сделать за $O(\sqrt{c})$, находя список делителей для каждого числа независимо. Для полного решения нужно воспользоваться более эффективным алгоритмом, например, для каждого числа от 1 до n перебрать все числа, которые на него делятся по аналогии с алгоритмом решета Эратосфена. Сложность решения будет $O(n \cdot ln(n))$. Существуют и более простые реализации полного решения задачи.

Отметим, что для значения $c = 4$ мы получим сразу два решения: $2 + 2 = 4$ и $2 \cdot 2 = 4$. Поэтому при $n \geqslant 4$ ответ надо уменьшить на 1.

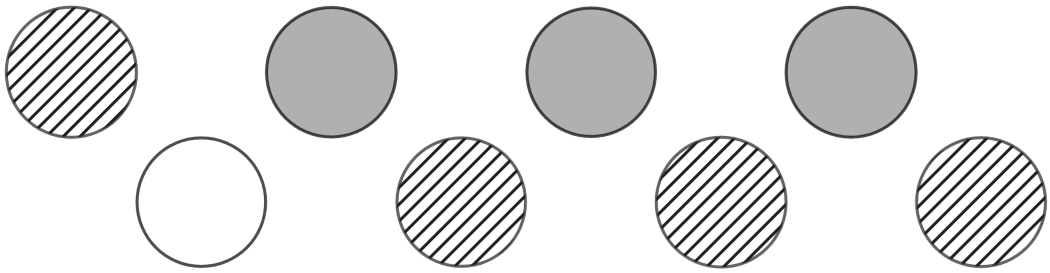
Задача D. Пила

Заметим, что можно сразу отсортировать данное множество чисел и взять наибольшие $\frac{n}{2}$ элементов на нечетные позиции, а остальные — на четные. При этом число на нечетной позиции всегда будет больше соседних чисел на четных позициях, за исключением случая равенства двух чисел.

Для решения задачи при $n \leqslant 16$ можно перебрать все подмножества чисел, которые мы будем изменять. Для каждого подмножества можно попробовать собрать пилообразную последовательность при помощи жадных алгоритмов.

Для эффективного решения задачи можно заметить следующее — последовательность является почти пилообразной тогда и только тогда, когда каждый элемент в ней встречается не более $\frac{n}{2}$ раз.

Действительно, даже если какой-либо элемент x встречается ровно $\frac{n}{2}$ раз, то мы можем расположить числа в требуемую последовательность следующим образом:



заштрихованные элементы равны x, серые элементы больше x, белый элемент меньше x

Несложно видеть, что если какое-то число x встречается больше $\frac{n}{2}$ раз, то два таких числа будут соседями и пилообразную последовательность собрать нельзя. Тогда для каждого числа достаточно посчитать, сколько раз оно встречается в исходном массиве и если какое-то число встречается $k > \frac{n}{2}$ раз, то ответ на задачу будет $k - \frac{n}{2}$, в противном случае мы можем собрать последовательность без изменений. Этот подход можно реализовать с асимптотической сложностью $O(n \cdot ln(n))$, например, воспользовавшись стандартным контейнером $std :: map$ (в C++).

Задача E. Оптимальное наложение

Для начала рассмотрим вторую подзадачу. Всего существует не более $O(n^2)$ различных наложений. Чтобы вычислить минимальный путь для конкретного наложения, применим метод динамического программирования, а именно посчитаем $dp[i][j]$ — величину минимального пути до клетки (i, j). Пересчитывать такую динамику можно следующим образом:

$$dp[i][j] = c[i][j] + min(dp[i-1][j], dp[i][j-1])$$

где $c[i][j]$ — это объединённая таблица. Также нужно аккуратно посчитать значения $dp[0][j]$ и $dp[i][0]$. В результате итоговая асимптотика для этой подзадачи составит $O(n^4)$.

Перейдём к решению третьей подзадачи. Для начала вычислим для каждой клетки таблицы B минимальный путь от верхней левой клетки, а для каждой клетки таблицы A — минимальный путь от нижней правой клетки. Теперь рассмотрим произвольное наложение и минимальный путь в нём. Этот путь равен сумме минимального пути до некоторой «краевой» клетки таблицы B и минимального пути до следующей клетки пути в таблице A. Таким образом, достаточно перебрать все пары: «краевая» клетка таблицы B и следующая клетка пути в таблице A. Общее количество таких пар составляет $O(n^3)$.

Для решения задачи на полный балл потребуется оптимизировать подход из предыдущей подзадачи. На самом деле для каждой краевой клетки таблицы B нужно найти оптимальную клетку перехода в таблице A, т.е. такую клетку, для которой расстояние от нижней правой клетки минимально. Это сводится к задаче поиска минимума в dp_A на прямоугольнике определённого вида. Для ускорения выполнения этой операции можно воспользоваться префиксными максимумами, рассчитанными из клетки $(1, n)$ и из клетки $(n, 1)$. В результате итоговая асимптотика решения составит $O(n^2)$.

Задача F. Пат ладьями

Для ответа на первую подгруппу можно написать полный перебор вариантов — для каждой из 4 фигур будет n^2 вариантов, где она может расположиться. Сложность может возникнуть при переборе вариантов, где король нападает на незащищенную ладью. Асимптотика $O(n^8)$.

Далее будем рассматривать общее решение. Для вариантов, когда король находится в углу — количество расположений нужно умножать на 4 (так как на доске 4 угловые клетки). Когда король

стоит у стенки — умножать на $4 \cdot (n-2)$. Когда король не находится в углу или у стенки – домножать на $(n-2) \cdot (n-2)$.

Для решения подгруппы $k = 2$ нужно рассмотреть 2 случая — первый, когда король находится в углу, и второй, когда король находится на краю доски. Варианта поставить пат двумя ладьями, когда король находится не у края — нет.



Рис.1 — $(n-2) \cdot (n-2)$

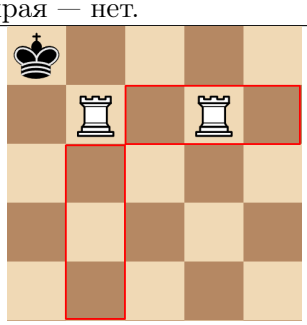


Рис.2 — $2 \cdot (n-2)$

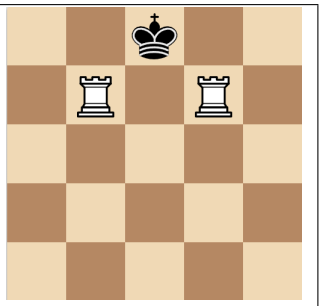


Рис.3 — 1

- Рисунок 1. Король в углу, ладья не под боем. Количество вариантов, куда поставить каждую ладью $(n-2)$, ладьи две, а значит общее количество в этом варианте — $(n-2) \cdot (n-2)$.
- Рисунок 2. Король в углу, ладья под боем. Одна ладья зафиксирована, а вторая может встать на любую клетку из красной области, а их — $2 \cdot (n-2)$.
- Рисунок 3. Король у стенки. В таком варианте есть единственное расположение ладей для пата.

Для решения при $k = 3$ рассмотрим сначала варианты, когда король в углу.

Рассмотрим случай, когда одна ладья под боем (её расположение можно выбрать единственным образом).

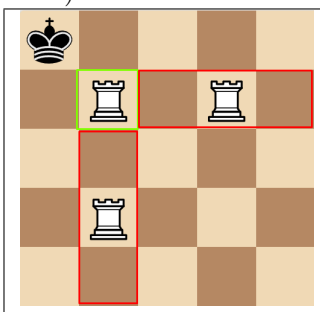


Рис.1 — $(n-2) \cdot (n-2)$

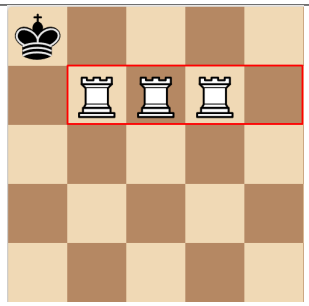


Рис.2 — $(n-2) \cdot (n-3)$

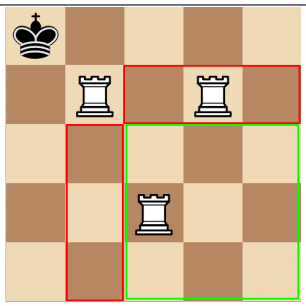


Рис.3 — $2 \cdot (n-2) \cdot (n-2)^2$

- Рисунок 1. Вторую ладью ставим на вертикаль, третью на горизонталь. У каждой по $(n-2)$ свободных клеток.
- Рисунок 2. Все ладьи на одной линии. Тогда количество способов расставить две оставшиеся ладьи — $C^2_{n-2} = (n-2) \cdot (n-3)/2$. Домножаем на 2, так как можно расположить так по вертикали и по горизонтали.
- Рисунок 3. Вторая ладья защищает первую $2 \cdot (n-2)$, а третья может расположиться на любой клетке зеленой области $(n-2)^2$.

Теперь ни одна ладья не будет под боем, а значит две ладьи должны перекрывать вертикаль и горизонталь. Первую и вторую ладью можно расставить $(n-2)$способами каждую:

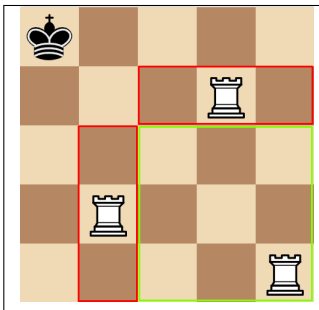


Рис.4 — $(n-2)\cdot(n-2)\cdot(n-2)^2$

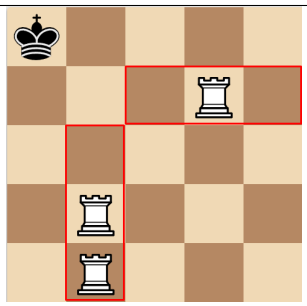


Рис.5 — $(n-2)\cdot(n-2)\cdot(n-3)$

- Рисунок 4. Третья ладья стоит в свободном зеленом квадрате $(n-2)^2$.
- Рисунок 5. Пусть третья ладья защищает вторую, тогда общее число способов расставить ладьи — $C^1_{n-2}\cdot C^2_{n-2} = (n-2)\cdot(n-2)\cdot(n-3)/2$. Домножаем на 2, так как третья ладья может защищать как вторую, так и первую.

Вариант, когда король у стенки:

Теперь ни одна ладья не будет под боем, а значит две ладьи должны перекрывать вертикаль и горизонталь. Первую и вторую ладью можно расставить $(n-2)$способами каждую:

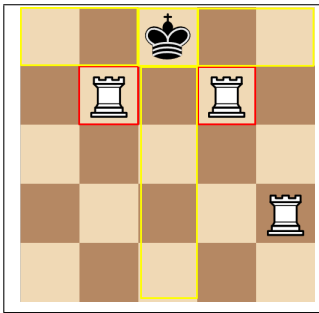


Рис.1 — $n^2-2\cdot n-1$

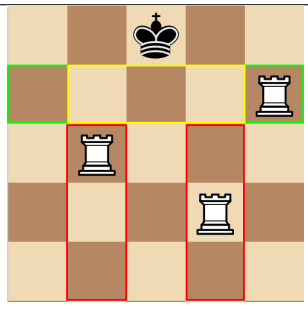


Рис.2 — $(n-2)\cdot(n-2)\cdot(n-3)$

- Рисунок 1. Первая и вторая ладья зафиксированы под боем. Третью можно расположить в любой незакрашенной клетке $n^2-(2\cdot n-1+2)$.
- Рисунок 2. Ни одна ладья не будет под боем. Тогда первую и вторую нужно расположить так, чтобы они перекрывали две боковые линии для короля, а третью чтобы она перекрывала горизонтальную линию. Первую и вторую ладью можно расставить $(n-2)$ способами каждую, а третью $(n-3)$ способами.

И разберём случаи, когда одна ладья под боем, способов расставить первую ладью — 2.

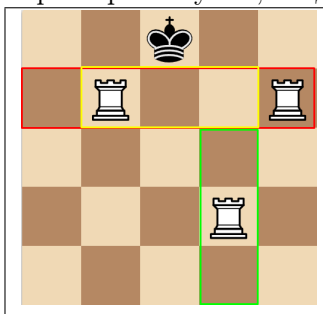


Рис.3 — $2\cdot(n-2)\cdot(n-3)$

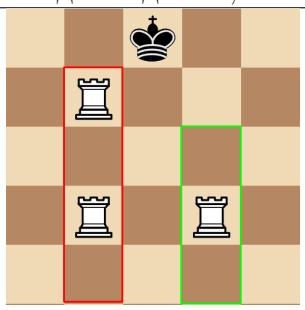


Рис.4 — $2\cdot(n-2)\cdot(n-2)$

- Рисунок 3. Пусть вторая ладья защищает первую на горизонтальной линии, тогда её можно расположить $(n-3)$ способами. А третья ладья должна перекрывать вторую вертикальную линию $(n-2)$ способами.
- Рисунок 4.Теперь вторая ладья защищает первую на вертикальной линии $(n-2)$ способами, а третью ладью всё также ставим перекрывать вторую вертикальную линию $(n-2)$ способами.

И последний случай, когда король находится внутри поля.

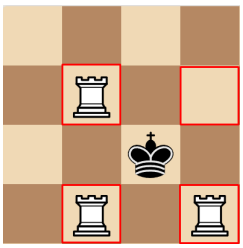


Рис.1 — 4

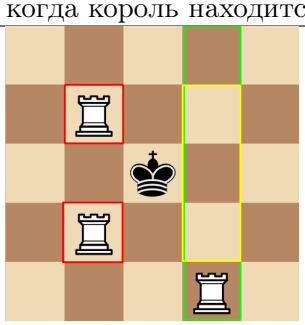


Рис.2 — $4 \cdot (n-3)$

- Рисунок 1. Расставим ладьи вокруг короля. Всего $C_4^3 = 4$ способа это сделать.
- Рисунок 2. Первую и вторую ладью можно выбрать единственным способом, а третью $(n-3)$ способами. Домножаем на 4, так как можно сделать повороты на 90 градусов.

Задача G. Бегуны и препятствия

Для того чтобы решить данную задачу было необходимо заметить, что в один момент времени с одной стороны в препятствие не может попасть 2 спортсмена одновременно. Доказательство этого факта остается в качестве упражнения читателю.

Далее необходимо понимать а в какие препятствия каждый спортсмен врезался. Для этого давайте хранить в каких позициях спортсмены начали гонку и в каком направлении они бегут. Тогда если появилось очередное препятствие в позиции x в момент времени t, то он должен будет стартовать в позиции $(x-t) \mod n$ (Все позиции мы считаем в 0-индексации). Тогда для этой позиции мы можем определить спортсмена и сказать, что он попал в это препятствие. И тогда нам остается пересчитать его движение в противоположном направлении. Тогда на самом деле такое столкновение спортсмена с препятствием равносильно его старту из позиции $(x+t) \mod n$. Аналогично мы можем сделать при движении против часовой стрелки. Данная часть решения может работать за $O(m \log m)$ от сортировки.

Теперь для каждого спортсмена мы знаем в какие препятствия он попадал. Далее давайте поймем, что нам не важно в какой позиции он попал в препятствие, а нам важен только сам факт попадания в него. Давайте представим, что наш спортсмен бежит не по кругу, а по прямой. Тогда в зависимости от времени между столкновениями с препятствиями будем поддерживать насколько далеко от 0 в обе стороны он смог убежать. Тогда если с расстояния отрезка до которого мы смогли добежать больше или равно n, то это значит, что мы пробежали весь круг. Тогда давайте искать ответ, например, бинарным поиском по времени сколько он будет бежать. Тем самым наше решение будет работать за $O((n+m)\log(T))$.