Задача A. Раскраска автобусов

В задаче требуется посчитать сумму максимальных показателей степеней двоек, на которые делятся числа с l по r. Для решения на 60 баллов достаточно пройтись по всем числам с l по r, и для каждого числа увеличивать показатель степени, пока она всё ещё делит число. Такое решение работает за $O((r-l)\log r)$, так как показатель степени не может быть больше логарифма числа.

Для решения задачи на полный балл давайте воспользуемся тем фактом, что ответ для отрезка (l, r) равен ответу для отрезка $(1, r)$ − ответ для отрезка $(1, l-1)$. Тогда, можно свести задачу к подсчёту ответа на отрезке $(1, x)$. Пусть f_i — количество чисел от 1 до x, кратных 2^i, а g_i — количество чисел от 1 до x, кратных 2^i, но не кратных 2^{i+1}. Заметим, что $g_i = f_i - \sum_{j=i+1}^{64} g_j = f_i - f_{i+1}$, а $f_i = \lfloor \frac{x}{2^i} \rfloor$. Из этих формул уже можно напрямую посчитать ответ, как $\sum_{i=1}^{64} g_i \cdot i$. Если расписать эту формулу: $\sum_{i=1}^{64} g_i \cdot i = \sum_{i=1}^{64} (f_i - f_{i+1}) \cdot i = \sum_{i=1}^{64} f_i = \sum_{i=1}^{64} \lfloor \frac{x}{2^i} \rfloor$, что посчитать уже проще. В итоге получаем решение за $O(\log r)$, набирающее 100 баллов.

Задача B. Муравьи

В решении на 60 баллов достаточно написать посекундную симуляцию происходящего за $O(K^2 \cdot T)$.

Заметим, что задачу можно независимо решать для муравьев двигающихся в горизонтальном и вертикальном направлении. Тогда, для каждой строки/столбца таблицы нам необходимо решить одномерную задачу. Заметим, что при столкновении два муравья не меняют своего относительного порядка. Пусть у нас есть строка/столбец длины L. Таким образом, если бы не было других муравьев, то за $2L$ времени один муравей вернулся бы на ту же позицию с тем же направлением. При столкновении можно считать, что муравьи просто обмениваются номерами (зрителю будет казаться, что муравьи не столкнулись, а просто продолжают двигаться дальше), то есть множество позиций/направлений всех муравьев в целом при столкновении сохраняется. Значит, с учетом сохранения относительного порядка, через $2L$ секунд все муравьи вернутся на те же позиции и будут иметь те же направления. Поэтому можно взять время T по модулю $2L$ и запустить моделирование. Получаем решение за $O(K^2(W+H))$, которое набирает 100 баллов.

Можно также было по формуле, учитывающей период $2L$, вычислить местоположение и направление какого то муравья через T секунд, а чтобы понять, какой это именно муравей, надо вспомнить, что муравьи не меняют своего относительного порядка и ответом является отсортированных массив вычисленных местоположений. Такое решение зависит только от количества муравьев и работает из $O(KlogK)$.

Задача C. Straight bet

В решении на 10 баллов за $O(n! \cdot n)$ можно просто перебрать все перестановки и посчитать количество ответов «`Больше! Меньше!`» на каждую из них, а затем найти сумму.

В более оптимальном решении нужно воспользоваться методом динамического программирования. Заметим, что во время отгадывания все, что нам нужно знать про текущее состояние, — это количество чисел, про которые еще есть смысл спрашивать и которые меньше чем x (обозначим их $cntL$), количество чисел из оставшихся в рассмотрении, больших чем x ($cntR$), и «`Больше!`» или «`Меньше!`» ($t = 0$ или 1) мы получили в ответ на последний вопрос (или же мы еще не спрашивали ничего). Тогда переход — это перебрать, про какое число мы будем задавать вопрос, посчитать, про сколько чисел теперь станет бесполезно спрашивать, и посчитать количество способов, которыми они могли стоять в последующей перестановке. Если мы хотели посчитать ответ для состояния $dp[cntL][cntR][t]$, и допустим мы выбрали i-е по величине число слева или справа в качестве загаданного, то у нас есть $\binom{cntL+cntR-1}{i-1} \cdot (i-1)!$ способов определить места и порядок выбывших чисел в оставшейся перестановке, а пересчитываем мы ответ через $dp[cntL-i][cntR][0]$ или $dp[cntL][cntR-i]$. Также, если мы раньше спросили про «`Больше!`», а теперь спрашиваем про «`Меньше!`», то нужно прибавить $(cntL+cntR-1)!$ к ответу. Таким образом, мы получили решение за $O(n^3)$, которое набирает 60 баллов.

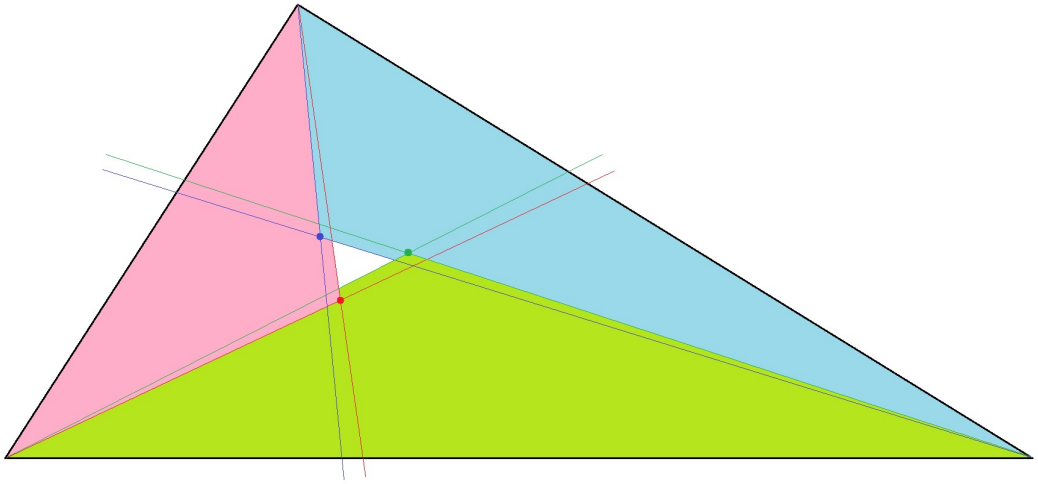
Чтобы оптимизировать решение до $O(n^2)$, рассмотрим формулу пересчёта подробнее: $\binom{cntL+cntR-1}{i-1} \cdot (i-1)! = (cntL+cntR-1)!/(cntL+cntR-i)!$. Тогда $dp[cntL-i][cntR][0] \cdot (cntL+cntR-1)!/(cntL+cntR-i)!$ можно разбить как $(cntL+cntR-1)!$ и

$dp[cntL - i][cntR][0]/(cntL + cntR - i)!$, а вторая часть не зависит от исходного состояния, поэтому на самом деле нам нужная некая префиксная сумма по всем $dp[x][cntR][0]/(x + cntR)!$ по x от $cntL - 1$ до 0. Эти префиксные суммы можно поддерживать походу, и тогда переход для состояния будет работать за $O(1)$, что и дает нам решение на 100 баллов.

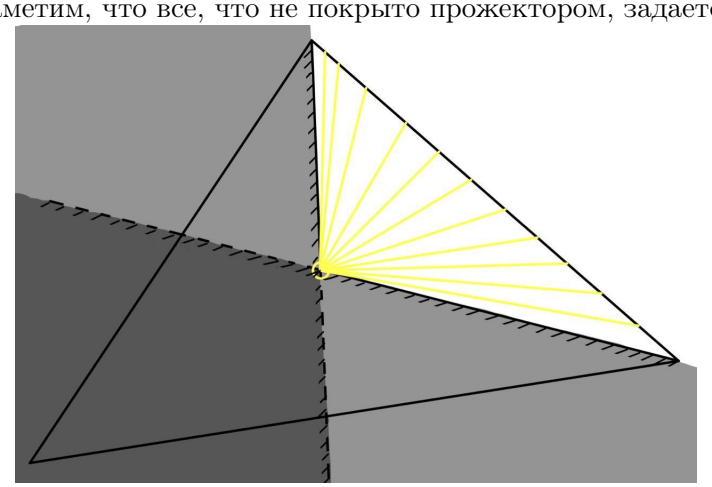
Задача D. 3 прожектора

Давайте в начале переберём 3! вариантов, какую стену какой столб будет освещать.

Каждый столб освещает участок в форме треугольника. Всего у нас есть три таких участка. Любые два участка имеют общую вершину и в окрестности этой вершины эти участки должны перекрываться, так как в противном случае останется часть, которая не будет покрыта ни одним из участков. Может показаться, что если каждая пара участков перекрывается, то весь треугольник будет ими покрыт. На самом деле, это условие является необходимым, но не достаточным. Ниже приведен контрпример из которого видно, что участки могут попарно перекрываться около общей вершины, но при этом не покрывать весь треугольник.



Так как же все-таки проверить, что вся площадь треугольника покрыта прожекторами? Для этого, давайте проверим, есть ли точка, непокрытая прожекторами, и лежащая внутри треугольника. Заметим, что все, что не покрыто прожектором, задается объединением двух полуплоскостей:



Переберём 2^3 вариантов выбора полуплоскостей для каждого прожектора, и рассмотрим эти 3 полуплоскости + полуплоскости задающие внутреннюю часть треугольника. Осталось проверить, что пересечение этих 6 полуплоскостей имеет ненулевую площадь. Это можно сделать с помощью алгоритма пересечения полуплоскостей(например за $O(n^3)$ - найдем все попарные точки пересечения; найдём те, что лежат внутри каждой из полуплоскостей; построим на них выпуклую оболочку и найдем ее площадь). Если хотя бы для одного варианта пересечение несчётно, то этот вариант распределения прожекторов не подходит, иначе — мы нашли ответ.

P.S. В этой задаче можно не искать площадь пересечения 6-и полуплоскостей, а только проверить пусто это пересечение или нет. Для этого можно воспользоваться трюком с тернарным поиском,

который ищет максимальное расстояние между верхней и нижней огибающей из полуплоскостей, при этом в явном виде они сами даже не строятся. Если это расстояние оказалось больше нуля, то пересечение не пусто.

Есть еще один способ посмотреть на решение задачи о поиске непокрытой точки. Если она существует, то она существует и в окрестности одной из точек пересечения внутри треугольника (включая его границу) упомянутых прямых, включая прямые, образованные сторонами треугольника. Можно перебрать все пары прямых, если они не параллельны, то найти точку их пересечения и, если она принадлежит треугольнику, то проверить точку вблизи нее в каждом из четырех углов, образованных пересекающимися прямыми.