Задача 1. «Кастинг»

Рассмотрим решение данной задачи отдельно для случаев поиска наибольшего и наименьшего количества актеров. В первом случае подходящий актер должен обладать всеми тремя свойствами в отдельности. Наибольшее количество актеров, обладающих всеми тремя свойствами, не превосходит каждое из чисел a, b, c. Несложно построить пример распределения свойств, при котором это количество в точности равно $\min\{a, b, c\}$. Например, пусть первым свойством обладают первые a актеров, вторым свойством – первые b актеров, а третьим – первые c актеров. Тогда всеми тремя свойствами обладают только первые $\min\{a, b, c\}$актеров.

При решении данной задачи в случае поиска наименьшего количества актеров рассмотрим два возможных способа рассуждений.

Первый способ заключается в следующем. Расставим актеров по кругу и будем последовательно назначать им свойства следующим образом: первым a актерам присвоим первое свойство, следующим за ними b актерам – второе, и следующим за ними c актерам – третье. При этом может оказаться, что некоторым актерам будет назначено два или три свойства. Поскольку движение осуществляется по кругу, то для того, чтобы какому-то актеру назначить три свойства, необходимо сначала сделать два полных круга, то есть назначить каждому актеру по два свойства. Количество актеров, которые получат три свойства, будет равно количеству актеров, которых обошли, делая третий круг. Если всего назначено $(a + b + c)$ свойств, а длина круга равна n, то за два круга будет назначено $2n$ свойств, а на третьем круге – оставшиеся $(a + b + c – 2n)$ свойств. При этом, если $(a + b + c – 2n) < 0$, то ни один актер не получит все три свойства. В этом случае ответ будет равен 0.

Теперь осталось только доказать, что если $(a + b + c – 2n) > 0$, то полученный вариант действительно будет соответствовать наименьшему количеству актеров. Для этого воспользуемся формулой включения-исключения для трех подмножеств A, B, C, соответствующих актерам, обладающим каждым из трех свойств:

$$|A| + |B| + |C| – |AB| – |BC| – |CA| + |ABC| = n.$$

Отсюда $|ABC| = n – a – b – c + |AB| + |BC| + |CA|$. Чтобы минимизировать $|ABC|$, нам нужно минимизировать $|AB| + |BC| + |CA|$. Минимальное значение $|AB|$ равно $(a + b – n)$, минимальное значение $|BC|$ равно $(b + c – n)$, минимальное значение $|CA|$ равно $(c + a – n)$. Подставляя эти значения в выражение для вычисления $|ABC|$, получаем требуемое соотношение:

$$|ABC| = n – a – b – c + (a + b – n) + (b + c – n) + (c + a – n) = a + b + c – 2n.$$

Второй способ рассуждений для случая поиска наименьшего количества актеров заключается в следующем. Сначала решим аналогичную задачу для двух свойств. Пусть a актеров обладают первым свойством, а b актеров – вторым. Тогда всего у нас имеется $(a + b)$ свойств на n актеров, следовательно, как минимум $(a + b - n)$ актеров обладают обоими свойствами. Пример получить несложно: выстроим актеров в ряд и наделим первым свойством первых a актеров, а вторым свойством – последних b актеров. При этом если $(a + b) \le n$, то актеров с обоими свойствами не будет вовсе. Поэтому ответ в задаче для двух свойств будет равен $\max\{a + b - n, 0\}$.

Теперь применим полученный результат к таким двум свойствам: 1) высокие и голубоглазые; 2) блондины. В первой группе не более $(a + b - n)$ актеров, во второй – c актеров. С учетом доказанного выше факта, всеми тремя свойствами будут обладать не более чем $\max\{\max\{a + b - n, 0\} + c - n, 0\} = \max\{\max\{a + b - n + c - n, c - n\}, 0\} = \max\{a + b + c - 2n, c - n, 0\} = \max\{a + b + c - 2n, 0\}$ актеров.

Последнее равенство верно в силу того, что $(c - n)$ не больше нуля.

Задача 2. «Города»

Одно из возможных решений данной задачи состоит из двух этапов. На первом этапе считываются входные данные и подсчитывается общее количество заданных городов, то есть букв ‘C’ во входном файле. Разделив это количество на 2, можно узнать требуемое количество городов K в каждом государстве.

На втором этапе выбираются города-клетки, относящиеся к первому государству. Это можно сделать разными способами. Самый простой из них заключается в следующем. Рассматриваем строки игрового поля сверху вниз, а элементы в каждой строке – слева направо и подсчитываем количество встречающихся городов. Процесс останавливается, как только встречается K-й город. Все пройденные к этому моменту города-клетки, включая клетку с K-м городом, следует отнести к первому государству, а остальные города-клетки – ко второму. Полученные государства будут связными, так как они состоят из нескольких последовательных полных строк и части еще одной строки, соседней с полными строками.

В простейшем случае, когда на игровом поле всего два города, для решения данной задачи достаточно найти один город и создать первое государство, состоящее только из клетки с этим городом. Второе государство будет состоять из всех остальных городов-клеток.

Задача 3. «A+B=C»

Для решения данной задачи можно применить метод динамического программирования [20]. С этой целью введем следующие обозначения.

Обозначим через *C'* число, состоящее из последних *k* цифр числа *C*.

Пусть s[*k*][*i*][*j*][0] – это количество способов разложения числа *C'* в сумму чисел *A'* и *B'*, каждое из которых состоит из *k* цифр, при этом *A'* начинается с цифры *i* ($0 \le i \le 9$), а *B'* – с цифры *j* ($0 \le j \le 9$). Рассматриваются также суммы, в которых слагаемые начинаются с нуля, например, 25 = 03 + 22.

Далее припишем слева к числу *C'* единицу и обозначим через *s*[*k*][*i*][*j*][1] количество способов разложения нового числа в такую же сумму, о которой шла речь выше (можно сказать, что здесь при сложении «переносится 1 в старший разряд»). Обобщая вышесказанное, *s*[*k*][*i*][*j*][*p*] – это количество способов представления числа *C'* в виде суммы с переносом *p* в старший разряд ($0 \le i \le 10$, $0 \le j \le 10$, $0 \le p \le 1$, $1 \le k \le |C|$ и $|C|$ обозначает длину исходного числа – *n*).

С учетом сказанного не сложно придти к выводу, что ответ на поставленную задачу будет равен сумме всех *s*[$|C|$][*i*][*j*][0] по всем *i* и *j*, отличным от нуля (слагаемые по условию не могут начинаться с нуля). Теперь осталось только вычислить *s*[*k*][*i*][*j*][*p*]. Покажем, как это можно сделать.

Пусть *t* – *k*-я справа цифра числа *C*. Рассмотрим три случая.

1) $(i + j) = (10p + t)$, то есть, $(i + j)$ равно либо *t*, либо $(10 + t)$ в зависимости от значения *p*. В этом случае переноса вправо не происходит, и нужно только просуммировать *s*[$k - 1$][*x*][*y*][0] для всех $0 \le x \le 9$ и $0 \le y \le 9$ кроме тех, для которых $i = x$ или $j = y$, то есть, две соседние цифры равны.

2) $(i + j) = (10p + t - 1)$. В этом случае 1 переносится в правый разряд, и теперь нужно просуммировать *s*[$k - 1$][*x*][*y*][1] (опять же кроме тех, для которых $i = x$ или $j = y$).

3) В остальных случаях *s*[$k - 1$][*x*][*y*][1] = 0, так как нельзя получить цифру *t* из чисел, начинающихся на *i* и *j*.

Алгоритм, реализующий описанное решение, имеет линейную сложность относительно длины числа *C*, но константа будет довольно большой: для каждой длины числа *k* необходимо вычислить $10 \cdot 10 \cdot 2 = 200$ чисел *s*[*k*][*i*][*j*][*p*], а для вычисления каждого из них требуется порядка $10 \cdot 10 = 100$ сложений. Но на самом деле большинство значений *s*[*k*][*i*][*j*][*p*] будет равно нулю (см. третий случай, о котором речь шла выше), и всего потребуется вычислить лишь порядка 40 ненулевых таких значений. В итоге, асимптотическая сложность алгоритма будет порядка $4000 \cdot N$.

Заметим, что искомое количество пар красивых чисел *A* и *B* может быть очень большим, поэтому ответ в задаче предлагается выводить по модулю (10^9+7). Все