Разбор задачи «Игра в напёрстки»

Зафиксируем за стаканами изначальную нумерацию. Рассмотрим функцию $p(i, j)$ — стакан с каким изначальным номером будет находиться на позиции i через j шагов. Тогда

- $p(0) = 0, 1, 2$
- $p(1) = 1, 0, 2$
- $p(2) = 1, 2, 0$
- $p(3) = 2, 1, 0$
- $p(4) = 2, 0, 1$
- $p(5) = 0, 2, 1$
- $p(6) = 0, 1, 2$

Таким образом, спустя 6 действий стаканы пришли в изначальное положение. Для решения задачи вычислим остаток $n\ mod\ 6$ и промоделируем соответствующее количество действий.

Разбор задачи «Игра «Банковские карты»»

Для начала научимся находить минимальное количество щелбанов, которое гарантированно получит Мориарти. Его цель — не проиграть как можно больше партий. Мориарти имеет смысл называть цифру `0` только когда Шерлок называет `0` — тогда Мориарти не проигрывает. Цифру `1` имеет смысл называть только против цифр `0` и `1`, цифру `2` против цифр `0`, `1` и `2` и так далее. Таким образом, Мориарти выгодно перебирать свои цифры от меньшей к большей и жадно использовать их для любых подходящих. Так он найдёт максимальное количество партий a, которые можно не проиграть, ответ — $n - a$.

Для поиска второй величины будем использовать аналогичный жадный алгоритм, только будем вычислять количество партий, которые можно выиграть. Цифра `0` оказывается бесполезной, цифра `1` бьёт `0` и так далее. Жадно перебираем цифры от меньших к большим и по максимуму бьём все подходящие цифры Шерлока.

Разбор задачи «Облако хештегов»

Данную задачу можно было решить разными способами. Один из вариантов решения — идти по строкам с конца, жадно стремясь оставить каждую очередную строку как можно более длинной.

Покажем это формально: заметим, что множество возможных длин каждой строки в корректном ответе образует отрезок от 1 до некоторой критической длины l_i. Действительно, если есть ответ, в котором i-я строка имеет длину $x \geqslant 2$, то есть также и ответ, в котором i-я строка имеет длину $x - 1$, так как можно все предыдущие строки просто сократить до одного символа. Таким образом, можно определить величину l_i — максимальную возможную длину i-й строки в корректном ответе.

Посчитаем l_i через l_{i+1}. Сократим $(i + 1)$-ю строку до длины l_{i+1} и рассмотрим два варианта. Во-первых, s_i может быть лексикографически не больше s_{i+1}, тогда, очевидно, можно положить $l_i = |s_i|$. Если же s_i больше s_{i+1}, то l_i никак не может быть больше $lcp(s_i, s_{i+1})$, где lcp обозначает длину наибольшего общего префикса двух строк (так как иначе в s_i будет длиннее любого префикса s_{i+1} вообще). В то же время, если сократить s_i до $lcp(s_i, s_{i+1})$, получится корректный ответ. Значит, можно смело положить $l_i = lcp(s_i, s_{i+1})$.

Заметим, что также по алгоритму построения, если мы сократим каждую i-ю строку до длины l_i, получится корректный ответ. В то же время, мы обязаны сократить i-ю строку хотя бы до длины l_i, значит, мы также получили правильный ответ на задачу.

Разбор задачи «Алёна и таблички»

Для каждой клетки (i, j) вычислим величину $up(i, j)$ равную максимальному r, такому что таблица неубывает по столбцу j если оставить строки с i по r включительно. Такую величину легко вычислить за время $O(nm)$:

- $up(i,j) = up(i+1,j)+1$, если $i < n$ и $a_{i,j} < a_{i+1,j}$;
- $up(i,j) = 1$ в противном случае.

Теперь для ответа на запрос (l_i, r_i) необходимо проверить, существует ли k, такое что $up(l_i, k) \geqslant r_i$. Для этого вычислим максимум в каждой строке $best(i) = \max\limits_{j=1}^{m} up(i,j)$.

Разбор задачи «Ханойская фабрика»

Сделаем следующее наблюдение: если какие-то два кольца i и j имеют равные внешние радиусы $b_i = b_j$, то их можно заменить на одно кольцо с таким же внешним радиусом, внутренним радиусом равным $min(a_i, a_j)$ и высотой равной $h_i + h_j$.

Используя данное наблюдение перейдём к задаче, в которой все внешние радиусы различны. Упорядочим кольца по убыванию внешнего радиуса — именно в таком порядке они будут идти в ответе. Теперь считаем, что $b_i < b_j$ для всех $i > j$. Для каждого кольца будем вычислять величину $ans(i)$ — максимальная высота башни, которая может заканчиваться именно i-м кольцом. Такая динамика легко вычисляется за время $O(n^2)$ по формуле $ans(i) = \max\limits_{j<i, a_j<b_i} ans(j) + h_i$.

Существует два различных подхода, позволяющих быстро вычислить все значения этой динамики:

1. Будем в отдельном массиве хранить кольца упорядоченными по внутреннему радиусу. Для всех колец с номерами в исходном (упорядоченными по внешнему радиусу) массиве меньше i запишем значение динамики, а для элементов с номерами больше либо равными i будем хранить 0. Тогда для вычисления оптимального перехода требуется взять максимальное на суффикс значение в данном массиве — с одной стороны только элементы $b_j > b_i$ будут иметь в нём ненулевые значения, с другой стороны массив упорядочен по a, поэтому мы можем взять суффикс элементов, для которых выполнится условие $a_j < b_i$. Для решения такой задачи подойдёт дерево отрезков или дерево Фенвика.

2. Заметим, что если $i < j < k$, кольцо k можно положить на кольцо j и кольцо k можно положить на кольцо i, то кольцо j можно положить на кольцо i. Действительно $a_i < b_k$ и $b_k < b_j$, следовательно $a_i < b_j$. Таким образом, достаточно для каждого i найти максимальное j, такое что $a_j < b_i$. Это можно сделать как уже упомянутыми выше структурами, так и используя более простой подход. Например, давайте двигаясь по массиву слева направо хранить в отдельном стеке индексы всех подходящих элементов в порядке возрастания. При добавлении нового элемента будем выбрасывать элементы с вершины стека до тех пор, пока не найдём подходящий. После этого сделаем вновь добавленный элемент последним в стеке. Данный алгоритм иллюстрируется следующим кодом:

```
stack <int> opt;
for (int i = 0; i < n; i++) {
     while (!opt.empty() && r[opt.back()].inner >= r[i].outer)
         opt.pop();
     if (!opt.empty())
         ans[i] = ans[opt.back()];
     ans[i] += r[i].height;
     opt.push(i);
}
```