Задача A. Отель «Континенталь»

Автор задачи и разработчик: Даниил Орешников

Для решения задачи достаточно научиться проверять, можно ли совместить два прямоугольника в один. Если хранить каждый прямоугольник как пару длин его сторон, то прямоугольники (a, b) и (c, d) могут быть совмещены в

- $(a, b + d)$, если $a = c$;
- $(a, b + c)$, если $a = d$;
- $(b, a + d)$, если $b = c$;
- $(b, a + c)$, если $b = d$.

Можно сократить перебор случаев, собирая прямоугольники $(a, b + c + d - a)$, если $a \in \{c, d\}$ и $(b, a + c + d - b)$, если $b \in \{c, d\}$.

Соответственно, для первой и третьей группы достаточно просто написать аккуратный перебор случаев, для остальных — по очереди перебирать прямоугольники и пытаться всеми способами присоединить очередной прямоугольник к каждому возможному варианту.

Количество возможных вариантов соединить прямоугольники может расти экспоненциально, однако прямугольники с равными сторонами считаются одинаковыми, поэтому достаточно хранить каждый вариант только один раз. Во второй группе тестов в принципе не может получиться больше одного варианта ответа, а в общем случае можно показать, что вариантов ответа может быть не более двух.

Действительно, если в конце можно получить какой-то прямоугольник, какая-то его сторона будет совпадать со стороной последнего добавленного прямоугольника. При этом его площадь фиксирована и равна сумме площадей всех прямоугольников, которые были использованы. При фиксированной площади одна сторона однозначно задает вторую, поэтому ответов может быть не более двух. Время работы решения — $\mathcal{O}(n)$.

Задача B. Противостояние фракций

Автор задачи: Александр Гордеев, разработчик: Константин Бац

Для решения первой подзадачи можно было явно попробовать сменить влияние одной группировки на другую в каждом городе, если это возможно. Среди всех получившихся распределений влияния, выделим те, которое являются нейтральными, а среди таких выберем такое, для которого нужно изменить главенствующую группировку в минимальном количестве городов. Удобнее всего написать такое решение при помощи рекурсии.

Во второй подзадаче между городами проходит ровно одна дорога. На самом деле, это означает, что города можно разбить на пары так, чтобы дороги были только между городами в одной паре. Это ограничение позволяет независимо для каждой пары соединенных городов проверить, можно ли получить нейтральное распределение в этой паре и за какое минимальное количество изменений влияния это можно сделать.

Третью подзадачу можно было решить, просто проверив, является ли заданное распределение нейтральным, поскольку главенствующие группировки изменять мы не можем. В четвертой подзадаче можно было изменить влияние группировки лишь в одном городе. Здесь можно было применить решение первой подзадачи, поскольку из-за данного ограничения, рекурсия будет иметь линейную от количества городов глубину, но ветвления в ней почти не будет.

Для полного решения необходимо обратить внимание на несколько фактов.

Заметим, что распределение влияния фракций может быть нейтральным только тогда, когда исходный граф можно раскрасить в два цвета так, чтобы никакие две соседние вершины не были покрашены в один цвет. Это классическая задача о двудольности графа. Покрасить граф в два цвета можно с помощью `dfs`, крася каждую следующую вершину в цвет, противоположный цвету предыдущей.

Если граф можно раскрасить в два цвета, то это можно сделать ровно двумя способами. Способы получаются друг из друга изменением цветов на противоположные. Соответственно, задача сводится к тому, чтобы независимо в каждой компоненте связности проверить ее двудольность и выбрать оптимальную из двух раскрасок.

Может получиться так, что раскраски не существует, потому что есть вершины, которые нельзя перекрашивать. Можно было в явном виде выписать обе раскраски каждой компоненты и проверить их корректность, а можно было заметить, что цвет любой вершины в КС однозначно задает цвета всех остальных вершин — тогда можно в КС, в которых есть вершины с фиксированным цветом, начать покраску из них и проверить отсутствие противоречий, а в остальных компонентах выбрать оптимальную из двух покрасок. Ответом на задачу будет сумма минимального числа перекрашиваний в каждой компоненте. Время работы решения — $\mathcal{O}(n+m)$.

Задача C. Маркер в библиотеке

Автор задачи: Даниил Орешников, разработчик: Константин Бац

В первой группе тестов можно просто перебрать все возможные способы последовательного выбора букв и выбрать минимальную из строк, которые можно получить.

Во второй и третьей подгруппах после небольшого анализа можно привести конструктивное решение. Если строка состоит только из букв ‘a’ и ‘b’, то можно сначала выписать самую правую букву ‘a’, затем выбрать самую правую из оставшихся в левой части, и так далее. В итоге все буквы ‘a’ будут выписаны до всех букв ‘b’, и это, очевидно, минимальная строка, которую можно получить. Если в строке при этом есть буквы ‘c’, то все еще можно выписать все буквы ‘a’ первыми, но надо аккуратно после этого восстановить порядок букв ‘b’ и ‘c’.

Для полного решения достаточно улучшить предыдущее решение. Можно заметить, что все еще можно выписать все буквы ‘a’ до остальных букв. Для этого стоит сначала разделить строку по самой правой букве ‘a’, затем по следующей, и так далее. После останутся отрезки строки между буквами ‘a’, которые надо таким же образом обработать в порядке слева-направо.

Таким образом, полный алгоритм заключается в повторении следующих действий:

1. достать из стека очередной отрезок строки;
2. найти на нем **самую правую из минимальных букв** и выписать ее;
3. разбить отрезок на два по этой букве и добавить их в стек.

Если искать минимальную букву за линейное время, получится решение за $\mathcal{O}(|s|^2)$. Более оптимально будет воспользоваться деревом отрезков или декартовым деревом, чтобы находить самый правый минимум за $\mathcal{O}(\log|s|)$, что дает решение за время $\mathcal{O}(|s|\log|s|)$.

Задача D. Группировки

Автор задачи и разработчик: Даниил Орешников

Упростим формулировку: требуется посчитать количество способов разбить дерево на группы размерами от 3 до k и отдельные вершины, чтобы в каждой группе была некоторая вершина вместе с множеством своих детей.

Первые две подгруппы решались полным перебором: можно было перебрать все разбиения n вершин и проверить, что разбиение удовлетворяет условию.

На бинарном дереве работает динамическое программирование по поддеревьям. Если некоторая вершина образует группу, то в этой группе обязаны быть оба ее ребенка. Таким образом, можно посчитать динамику, аналогичную динамике для поиска максимального независимого множества в дереве, только в данном случае в роли вершины могут выступать отдельные вершины или тройки из родителя и двух детей.

Остальные подгруппы расчитаны на разные версии полного решения, в которых менее или более оптимальным образом вычисляются значения динамики. Приведем сразу полное решение. Будем считать значения $\texttt{dp}[v]$ — количество способов разбить указанным образом поддерево вершины v.

Разобьем его на $\texttt{dp}[v] = \texttt{dp}_0[v] + \texttt{dp}_1[v]$, где 0 или 1 означает флаг принадлежности самой вершины v некоторой группе.

Посчитать $\texttt{dp}_0[v]$ просто: если v не принадлежит никакой группе, то можно независимо разбить все поддеревья ее детей, поэтому

$$\texttt{dp}_0[v] = \prod_{u_i \in \mathrm{ch}(v)} \texttt{dp}[u_i].$$

Чуть сложнее искать $\texttt{dp}_1[v]$. Если v выбрана в некоторую группу, то вместе с ней в группе должны быть от 2 до $k-1$ ее детей. Решим для начала случай $k = 3$. Можно просто перебрать все способы выбрать двух детей и записать ответ как

$$\texttt{dp}_1[v] = \sum_{i \neq j} \left(\texttt{dp}_0[u_i] \cdot \texttt{dp}_0[u_j] \cdot \prod_{t \neq i, t \neq j} \texttt{dp}[u_t] \right).$$

Такой пересчет займет в сумме $\mathcal{O}(n^2)$ времени и пройдет пятую подгруппу.

Соптимизируем решение до $\mathcal{O}(nk)$. Будем в каждой вершине считать $\texttt{sub}[t]$ — количество способов выбрать ровно t детей в группу к родителю, а поддеревья остальных детей разбить на группы независимо друг от друга. Будем перебирать детей по i от 1 до $\mathrm{ch}(v)$ (количества детей) и делать пересчет

$$\texttt{sub}[t] \leftarrow \texttt{sub}[t] \cdot \texttt{dp}[u_i] + \texttt{sub}[t-1] \cdot \texttt{dp}_0[u_i] \text{ по всем } t,$$

то есть либо взять t детей из предыдущих, либо взять $t-1$ детей из предыдущих и добавить текущего.

В конце достаточно добавить к $\texttt{dp}_1[v]$ соответствующие значения $\texttt{sub}$. Ответ на задачу будет лежать в $\texttt{dp}[1]$.