Задача A. Memory snow

Для решения задачи на 20 баллов за $O(n^3)$ достаточно перебрать подстроку исходной строки длины $\geqslant 3$ и проверить, что в ней ровно одна буква S или U линейным проходом.

В решении на 50 баллов за за $O(n^2)$ достаточно перебрать левый конец подстроки, увеличивать правый конец и поддерживать счётчик количеств каждой буквы.

Для решение задачи на 100 баллов за $O(n)$ заметим, что в любой подходящей подстроке ровно одна буква встречается ровно один раз(так как длина хотя бы 3), значит, можно ее перебрать, и посчитать количество корректных подстрок, которые ее содержат. Для этого давайте предпосчитаем для каждой буквы ближайшие слева и справа вхождения такой же буквы простыми линейными проходами, тогда исходная подстрока должна лежать между ними.

Пусть мы рассматриваем букву на позиции i в нумерации с нуля, левое вхождение на позиции l_i(или же -1, если слева этой буквы нет), правое вхождение на позиции r_i(или же n), тогда существует $(i - l_i) \cdot (r_i - i)$ подходящих строк, из них нужно вычесть количество строк длины $\leqslant 2$, которые лежат в этом интервале и содержат позицию i.

По другому нужные величины можно получить за один проход "сжатием строки-— заменой блока из повторяющихся букв на их количество. Сжатая строка может выглядеть, например, так:

5 1 3 5 1 2 7 1

Нас в этой строке интересуют значения, равные единице, и соседние с единицами значения (единицы в крайней левой или крайней правой позициях нужно обработать отдельно). Если же рядом стоят два блока, состоящие больше чем из одной буквы каждый, то к ответу нужно добавить последовательности состоящие из крайней буквы одного блока и как минимум двух букв соседнего блока.

Задача B. Микроконтроллеры

Заметим, что две дорожки на одной стороне от платы нельзя провести, если они пересекаются, но не являются вложенными: (a, b), (c, d) и $a < c < b < d$ или $c < a < d < b$. Тогда любые две дорожки на одной стороне должны быть либо вложены, либо не пересекаться.

Упорядочим дорожки от микроконтроллера с меньшим номером к микроконтроллеру с большим $(a, b), a < b$, и скажем, что на позиции a мы поставим $(_i$, а на позиции b — $)_i$, тогда мы должны получить две правильные скобочные последовательности с k типами скобок.

Тогда за $O(N^2)$ задачу можно решать так: переберём ответ, проверим за линию со стеком, что микроконтроллеры можно соединить, то есть, что сверху и снизу получается правильная скобочная последовательность.

Для решения задачи за $O(NlogN)$ достаточно сделать бинпоиск по ответу, вместо его перебора, так как если мы смогли соединить первые k микроконтроллеров, то и меньшее их количество мы можем соединить.

Существует также линейное решение: можно точно также поверять, что сверху и снизу у нас правильные скобочные последовательности, но как только мы встретили конфликт (например, сейчас идёт скоба) типа j, тогда как в стеке последняя скобка (типа i, мы говорим, что i и j не могут лежать оба в ответе, значит более поздний из них умирает, и ответ точно меньше $max(i, j)$. Затем мы либо удаляем i из стека(и из дальнейшего рассмотрения), если $i < j$, либо пропускаем j. Когда мы удаляем набор скобок с временем появления t, все пары с большим временем также удаляются. Таким образом, мы можем найти ответ за два прохода за $O(N)$ со стеком, и гарантированно получить 100 баллов.

Задача C. Миллион алых роз

В решении на 30 баллов достаточно перебрать все подпоследовательности и посчитать количество различных, например с помощью $std::set$ за $O(2^n \cdot n^2 \cdot \log n)$

В решении на 60 баллов будет хранить динамику вида — dp_i, сколько различных подпоследовательностей заканчивается на элементе i. Чтобы не учитывать ничего дважды, давайте учитывать подпоследовательность, только если она максимально сдвинута вправо: не должно быть элемента после последнего, с таким же значением, между последним и предпоследним не должно быть элемента со значением предпоследнего элемента и т.д. (то есть для каждого значения в динамике мы будем иметь ввиду его последнее вхождение).

Тогда, для пересчёта динамики в i элементе: $dp[i] = 1 + dp[j_1] + dp[j_2] + \ldots$, где $j_1, j_2, \ldots$ — позиции последних вхождений каждого элемента. Ответом на задачу будет сумма $dp[i]$ по всем i таким что, справа от i не существует элемента с таким же значением. Получаем решение за $O(N^2)$.

Для улучшения решения до линейного, нам достаточно эффективно хранить сумму $dp[j_1] + dp_[j_2] + \ldots$ по последним вхождениям каждого уникального значения. Эту сумму легко обновлять, если хранить по каждому числу текущее значение динамики, и при встрече нового числа вычитать старое и прибавлять новое, что даёт решение за $O(N)$.

Альтернативное решение использует схожие идеи. Пусть $dp[i]$ — количество различных подпоследовательностей, использующих только первые i элементов массива, включая пустую. Тогда ответ — $dp[N] - 1$, а $dp[0] = 1$. Для пересчёта динамики заметим, что на первых i элементах все подпоследовательности, это либо подпоследовательности на первых $i - 1$ элементах, либо они же, но с дописанным i элементом. Но, если i элемент уже встречался раньше на позиции j, то мы учтём дважды часть подпоследовательностей, а именно, те, которые состояли из чисел до позиции j, то есть их $dp[j-1]$. Поэтому, $dp[i] = (dp[i-1] \cdot 2 - dp[j-1])$, где j — предыдущее вхождение элемента на позиции i.

Обратите внимание, что все действия необходимо проводить по заданному модулю M, при этом к операциям вычитания (в том числе единицы для получения окончательного ответа) нужно добавить число M, до взятия по модулю, чтобы гарантировать неотрицательность выражения на C++.

Задача D. Memento

В группах с $m \leqslant 1000$ можно предподсчитать состояния массива после каждой операции, посчитать на них частичные суммы квадратов, и отвечать на каждый запрос за m взятий сумм на отрезке. Такое решение набирало 30 баллов, и имело сложность $O((n+q) \cdot m)$.

В группе с $q = 1$ нужно научиться быстро обрабатывать изначальные m операций, и поддерживать запрос взятия суммы квадратов на всём массиве. Это можно делать с помощью дерева отрезков, в котором будем хранить как сумму квадратов на отрезке, так и сумму этих элементов на отрезке. Если каждый элемент на отрезке, состоящем из k элементов увеличивается на x, то сумма квадратов меняется на $k \cdot x^2 + 2x \cdot S$, где S — сумма элементов на этом отрезке, поэтому, зная сумму, мы легко можем пересчитать и новую сумму квадратов.

На полный балл легче всего было сдать решение с корневой декомпозицией по изначальным операциям. Давайте выразим каждый из q запросов через два: по всем операциям до y-й включительно и по первым $x - 1$ операциям, тогда нам нужно просто уметь считать сумму всех версий чисел на отрезке с l по r после первых k операций.

При обработке очередного блока операций, рассмотрим как они разбивают массив на корень подотрезков, чтобы каждый из них либо целиком лежал в операции, либо целиком не лежал. Для каждого подотрезка будем хранить сумму чисел, историческую сумму чисел и историческую сумму квадратов. При обработке очередной операции будем их вручную пересчитывать для каждого подотрезка отдельно, это можно сделать за $O(1)$. Ответ на запрос можно также пересчитать за $O(n^{0.5})$ для каждого отрезка отдельно, если ещё хранить префиксные суммы на обычных числах, квадратах, исторической сумме чисел и сумме квадратов, и пересчитывать их после каждого блока операций. В итоге получаем решение за $O(n^{1.5})$, которое набирало 100 баллов при достаточно эффективной реализации.