

# Выход участников

Автор задачи: Алексей Михненко

Разработчик задачи: Артём Агафонов

Будем симулировать процесс. Переберём участников в порядке возрастания номеров. Если цвет очередного участника совпадает с цветом предыдущего вышедшего, то поместим его в очередь. Иначе отметим, что данный участник выходит следующим, а затем выпустим участника, ожидающего в очереди, если она не пуста. Наконец, когда мы переберём всех участников, выпустим всех участников, которые ожидают в очереди. Получаем линейную симуляцию данного процесса, а значит данное решение будет работать за  $O(qn)$ .

Заметим, что если при такой симуляции очередь пуста и цвет футболки участника отличается от цвета предыдущего вышедшего, то данный участник выйдет под своим номером. Заметим, что все участники, имеющие номер меньше данного, не влияют на расстановку его и всех последующих, а значит мы можем считать, что на таких участниках процесс начинается заново с поправкой на их номер.

Пусть мы начинаем процесс с участника номер  $i$ , как найти следующего участника, с которого можно считать, что процесс начался заново? Поставим  $+1$  на позиции участников с цветом футболки  $a_i$  и  $-1$  на все остальные. Тогда наименьший индекс  $j$  такой, что сумма от  $i$  до  $j$  включительно равна  $0$ , будет искомым. Для каждой позиции  $i$  проведём ориентированное ребро в  $p_i = j$ . Такие ребра образуют ориентированный лес.

Поймём, как с помощью данного леса можно отвечать на запрос. Начиная с первого участника, будем постепенно переходить по рёбрам пока следующий участник имеет номер не больший, чем  $y$ . Пусть мы остановились на  $i$ , тогда по свойству выше можем считать, что процесс начинается с участника  $i$ . Заметим, что при расстановке участников на отрезке с  $i$  по  $p_i - 1$  на позициях  $i$ ,  $i + 2$ ,  $i + 4$ , ...,  $p_i - 1$  будут стоять участники с цветом футболки  $a_i$ , а на позициях  $i + 1$ ,  $i + 3$ , ...,  $p_i - 2$  все остальные. Тогда, чтобы узнать позицию участника  $y$ , нужно выяснить сколько участников с цветом  $a_i$  стоят до него. Для этого можно воспользоваться бинарным поиском по массиву позиций участников с футболками цвета  $a_i$ . Такой массив можно легко поддерживать при запросах изменения.

Далее, задача разбивается на две части: необходимо динамически поддерживать данный лес при смене цветов двух соседних участников и быстро находить ребро, накрывающее позицию  $y$ . Ограничения в некоторых подгруппах позволяли упростить реализацию данных частей решения. Далее, будет рассмотрено решение на полный балл, использующее link-cut. Другой вариант – использование корневой декомпозиции, которая при аккуратной реализации тоже позволяла набрать полный балл в данной задаче.

Вторая часть, состоящая в поиске накрывающего ребра, реализуется операцией expose от вершины  $1$ , что позволяет выделить путь, в котором будут лежать все участники, на которых процесс будет начинаться заново. Найти интересующего нас участника можно с помощью спуска в полученное splay-дерево.

Если мы выясним, у каких участников и как меняются значения  $p_i$ , то данные изменения можно поддержать в дереве с помощью операций link и cut. Заметим, что для цветов, не равных  $a_x$  и  $a_{x+1}$ , оба участника вносили вклады, равные  $-1$  в соответствующие суммы. Значит их перестановка не повлияет на суммы. Если цвета  $a_x$  и  $a_{x+1}$  совпадают, то никакого изменения не происходит. Рассмотрим случай  $a_x \neq a_{x+1}$ . Во-первых, изменятся рёбра исходящие из  $x$  и  $x + 1$ . Заметим, что для цвета  $a_x$  произошло изменение вкладов позиций  $x$  и  $x + 1$  с  $+1$ ,  $-1$  на  $-1$ ,  $+1$ . Теперь, при наборе суммы из какой-то позиции  $i$  цвета  $a_x$  алгоритм может получить  $0$  на позиции  $x$ . Это означает, что ранее сумма до позиции  $x$  была равна  $2$ . Зная это мы можем найти позицию, в которой префиксная сумма для цвета  $a_x$  оказалось на  $2$  ниже, чем в  $x$ . Например, это можно реализовать спуском в дерево отрезков для цвета  $a_x$ , которое в листе  $i$  хранит сумму на префиксе до  $i$ -го участника с футболкой цвета  $a_x$  (заметим, что хранить такие суммы достаточно, так как между такими участниками префиксные суммы линейно убывают). Данная позиция и будет той, в которую попало бы ребро, которое теперь мы должны перенаправить в  $x$ . В силу способа проведения рёбер такое ребро будет одно. Аналогично, можно разобрать случай для цвета  $a_{x+1}$  – из него мы тоже получим

не более одного кандидата, для которого изменится значение в массиве  $p$ . Чтобы найти позицию, в которую будет смотреть новое ребро, необходимо сделать ещё один запрос в соответствующее дерево отрезков, чтобы найти первое вхождение после позиции  $i$  префиксной суммы на 1, меньшей, чем префиксная сумма до позиции  $i$ .

Таким образом, получим решение, работающее за  $O(q \log(n))$ . Также, на полный балл можно было реализовать решения, работающие за  $O(q \log^2(n))$ , в которых splay-дерево в link-cut заменяется на декартово дерево, или за  $O(q\sqrt{n})$ , в котором весь массив разбивается на блоки размера  $\sqrt{n}$  и для каждого элемента поддерживается сжатый переход, который указывает на первый элемент, лежащий вне данного блока, в который можно перейти, двигаясь по ребрам вверх. Тогда при запросе изменения нам потребуется обновить данный переход для элементов из не более чем 4 блоков, а поиск накрывающего ребра будет выполняться проходом по таким сжатым переходам до тех пор, пока не будет достигнут блок, в который попал запрос.