Разбор задачи «Рейнджеры в автобусе»

Будем обрабатывать пассажиров по очереди и для каждого определять, кем он мог бы быть. Сразу заметим, что розовый рейнджер может есть куда угодно, поэтому каждый пассажир мог бы быть розовым.

Решение первой подзадачи (25 баллов): для каждого места будем помнить, свободно ли оно (массив bool длины $2n$). Проверим, мог бы очередной пассажир быть красным рейнджером. Для этого найдём место, на которое сел бы красный. Переберём циклом ряд от 1 до n, если попался ряд, в котором есть свободное место, выбираем левое, если оно свободно, или правое, если нет — это и есть место, куда сел бы красный. Поскольку он выбирает место однозначно, мы можем определить, мог бы этот пассажир быть красным — нужно проверить, что он сел именно на это место. Таким же образом узнаем, куда сели бы синий, жёлтый и чёрный – разница будет в порядке перебора рядов (от 1 до n или наоборот) и в порядке выбора места в ряду (сначала левое или правое). После обработки пассажира отметим его место как занятое.

Решение работает за $O(nk)$ и использует $O(n)$ памяти.

Решение второй подзадачи (50 баллов): на самом деле предыдущее решение работает за $O(k^2)$ времени. Дело в том, что каждый цикл по рядам останавливается, когда находит ряд со свободным местом — а количество занятых рядов не более $k/2$. Однако оно использует $O(n)$ памяти, а это слишком много для второй подзадачи. Есть два способа решения этой проблемы:

- Вместо массива использовать std::set, в котором хранить занятые места.
- Хранить только $k/2 + 1$ первых рядов и столько же последних.

Тогда решение будет использовать $O(k)$ памяти.

Полное решение (100 баллов): вспомним, что каждый из циклов останавливается, когда находит ряд, в котором есть свободное место. Это означает, что нужно быстро находить первый и последний незанятые ряды. Будем поддерживать эти значения — $first$ и $last$. Изначально $first = 1$, $last = n$. После каждой итерации цикла обновим эти значения. Будем увеличивать $first$ на 1, пока ряд $first$ занят, затем уменьшать $last$ на 1, пока ряд $last$ занят. Занятых рядов не может оказаться больше $k/2$, поэтому $first$ и $last$ сделают $O(k)$ шагов. Таким образом, решение работает за $O(k)$.