

## Разбор задачи «Расписание»

В первую очередь следует формализовать задачу. Лекции — отрезки на прямой. Если известно, на какие лекции Арсений пойдёт, то результатом будет длина их объединения. Итак, дано  $N$  отрезков на прямой, требуется выбрать не более  $K$  из них с максимальной длиной объединения.

Для предобработки и для основной части решения полезно отсортировать отрезки по возрастанию начала (а при равенстве — по убыванию конца). Такой порядок позволяет и подготовить данные, и естественным образом решать задачу.

Теперь исходные данные можно упростить: ясно, что отрезок  $i$  вложен в отрезок  $j$  ( $l_j \leq l_i \wedge r_i \leq r_j$ ), то отрезок  $i$  в оптимальном решении не пригодится. Избавиться от вложенных отрезков можно так: проходим массив в отсортированном порядке, удаляя все отрезки, чей конец находится левее конца последнего неудаляемого. После такой обработки, в дальнейшей части решения потребуется для каждого отрезка  $i$  найти ближайший справа ( $j > i$  после сортировки), не пересекающийся с данным (назовём его  $next_i$ ).

Вычисление  $next_i$  можно выполнить *методом двух указателей* за линейное время. Создадим указатели  $i$  и  $j$ , пусть вначале они установлены на 1-й элемент. На каждой итерации будем сдвигать указатель  $j$  вперёд, пока  $r_i \geq l_j$ . После этого присвоим  $next_i \leftarrow j$  и увеличим  $i$  на 1. Несложно видеть, что каждый из двух указателей будет инкрементирован (увеличен на единицу) не более  $N$  раз, таким образом время выполнения этой части алгоритма есть  $O(N)$ .

Остальная часть решения основана на методе *динамического программирования*<sup>1</sup>. Параметрами выберем следующее: количество уже выбранных отрезков (от 0 до  $K$ ), номер текущего рассматриваемого отрезка, а также логическое значение: выбран ли текущий отрезок. Функцией от состояния будет длина объединения уже взятых отрезков, её требуется максимизировать.

Переходы между состояниями зависят от третьего параметра. Если текущий отрезок не выбран, то можно либо перейти к следующему, не выбирая его, либо выбрать текущий. Если же текущий отрезок выбран, также есть два перехода. Первый — выбрать ещё один отрезок, пересекающийся с текущим. В таком случае имеет смысл выбрать только отрезок с номером  $next_i - 1$  (если  $next_i > i + 1$ ). Второй — перейти к ближайшему незадетому отрезку и не выбирать его (это и есть  $next_i$ ).

Относительно высокие ограничения могли создать «подводные камни» разного рода: например, матрица, хранящая все результаты для всех наборов параметров, имеет около  $10^5 \times 100 \times 2 = 2 \times 10^7$  элементов и занимает  $2 \times 10^7 \times 4 = 80 \times 10^6$  байт. Это не превышает ограничения по памяти, однако может негативно сказаться на времени выполнения. Также большие объёмы данных не всегда успешно кэшируются. Чтобы преодолеть эту проблему, достаточно было хранить только два столбца вычисляемой матрицы (один соответствует текущему количеству выбранных отрезков, второй — на единицу большему). Объём используемой памяти сокращается в 50 раз.

---

<sup>1</sup>Если вы не знакомы с этим термином, стоит обратиться, например, к книге Томаса Кормена, Чарльза Лейзерсона и Рональда Ривеста «Алгоритмы: построение и анализ». Или попробуйте разобраться с ним самостоятельно на рассматриваемом примере.