

Фабрика эхо

Автор задачи: Даниил Орешиников, разработчик: Павел Скобелкин

Давайте воспользуемся декартовым деревом для решения задачи. Будем поддерживать декартово дерево по неявному ключу из текущих сегментов, каждый из которых также будет декартовым деревом по неявному ключу, в котором будем хранить сами элементы очереди в нужном порядке. Научимся обрабатывать запросы всех типов:

1. Чтобы добавить элемент в конец очереди, нужно достать последний сегмент из дерева и добавить в его конец новый элемент;
2. Заметим, что запросы удаления можно обрабатывать «наивно», ведь суммарно удалений произойдет не больше количества добавленных элементов. То есть для обработки этого запроса нужно пройти по всем сегментам и удалить первый элемент из каждого, после этого удалить пустые сегменты;
3. Чтобы разделить сегмент, достаточно разделить декартово дерево по неявному ключу для данного сегмента на два дерева, и вставить новый сегмент сразу за разделенным;
4. Запрос объединения всех сегментов также можно выполнять «наивно», ведь их суммарно будет не более, чем запросов разреза. То есть для обработки этого запроса просто объединим все сегменты в один.

Таким образом эту задачу можно решать с помощью вложенных ДД.

Также есть решение, не использующее декартово дерево или хотя бы два вложенных ДД. Давайте хранить все элементы очереди в обычном массиве и помечать уже использованные элементы.

Тогда сегменты можно хранить в виде встроеного дерева поиска (например, `ordered_set` в C++, или также в декартовом дереве, в данном решении оно будет одномерное). Каждый сегмент будет задаваться парой чисел: индекс его начала в текущем массиве и количество элементов в нем. Тогда решение остается таким же, но нужно по другому обрабатывать запрос номер три.

Теперь тяжело искать индекс начала второго сегмента, если мы разрежем сегмент после i -го элемента. Заметим, что тогда индекс начала нужного сегмента будет индексом $i + 1$ -й единицы, начиная с индекса начала текущего сегмента. Этот запрос, например, можно обрабатывать техникой спуска по дереву отрезков. Тогда если поддерживать использованные элементы в дереве отрезков, то мы сможем эффективно отвечать на все запросы.