1 Дерево отрезков

1.1 RMQ, RSQ, решения offline

Для начала рассмотрим две классические задачи, для решения которых в дальнейшем и применим дерево отрезков.

В задаче RSQ (Range Sum Query, запросы суммы на отрезках) требуется поддерживать последовательность a_i и выполнять запросы двух видов:

- сообщить сумму $a_i + a_{i+1} + \ldots + a_j$,
- изменить элемент a_i.

Можно решать её «в лоб», поддерживая массив и вычисляя сумму каждый раз с начала до конца. Сложности запросов в таком случае составит $O(n)$ и $O(1)$ соответственно.

Можно подсчитать частичные суммы: $s_i = \sum_{j=1}^{i} a_j$. Это позволит вычислять сумму с i-го по j-й элемент за $O(1)$. Однако, изменение i-го элемента потребует пересчёта всех последующих частичных сумм, то есть будет работать за $O(n)$.

Придумав два «крайних» решения, перейдём к более сбалансированным.

Выберем число k и разобьём нашу последовательность на части по k элементов (лишние элементы можно будет отнести к последней части).

Подсчитаем суммы $p_i = \sum_{j=1}^{k} a_{k(i-1)+j}$. Теперь ответ можно вычислить как сумму некоторых p_i для тех частей, которые полностью входят в интервал запроса, а также a_j, которые остались непокрытыми после этого. Изменение элемента в данном случае повлечёт изменение также одного значения p_i. Так мы получаем решение с оценками времени $O(\sqrt{n})$, $O(1)$ (докажите).

Аналогично можно построить решения для задачи о поиске минимума на отрезке, и вообще любой ассоциативной функции. Заметим, что при отсутствии возможности обращать эту функцию (как в примере с минимумом) частичные суммы придётся пересчитывать полностью.

1.2 Общая концепция дерева отрезков

Продолжим идею с разбиением последовательности на части. А именно, будем строить двоичное дерево следующим образом:

- Пусть рассматривается последовательность $a_0, a_1, \ldots, a_{n-1}$

- Корню сопоставим полуинтервал $[0, n)$
- Далее рекуррентно будем сопоставлять детям вершины с полуинтервалом $[l, r)$ полуинтервалы $[l, m)$ и $[m, r)$, где $m = \left\lfloor \frac{l+r}{2} \right\rfloor$
- Если полуинтервал состоит из одного элемента, детей у вершины не будет

В каждой вершине запишем результат выполнения нужной операции — сумму, минимум или другую, — для соответствующего полуинтервала.

1.3 Операция обновления элемента

При обновлении элемента изменяется значение в полуинтервале, который ему соответствует, а также во всех полуинтервалах, которые его содержат. Несложно заметить, что с точки зрения дерева это все вершины, лежащие на пути от соответствующей вершины до корня.

1.4 Операция запроса на отрезке

Пусть нужно вычислить ответ на полуинтервале $[a, b)$. Будем выполнять следующую рекурсивную функцию:

- Пусть текущая вершина дерева v соответствует полуинтервалу $[l, r)$
- Если $[a, b)$ не пересекается с $[l, r)$, полуинтервал $[l, r)$ учитывать не нужно — нужно вернуть нейтральный элемент относительно операции (для суммы это ноль, для минимума это $+\inf$)
- Если $[a, b)$ полностью покрывает $[l, r)$, полуинтервал $[l, r)$ нужно учесть полностью — учтём его, выполнив нужную операцию (в простейшем случае сумму) со значением, хранящемся в вершине v, после чего завершим функцию
- Во всех остальных случаях запустим функцию рекурсивно от детей v и выполним операцию для полученных значений

1.5 Оценка времени работы

Докажем, что время выполнения одного запроса не превосходит $O(\log n)$.

Пусть из вершины был произведён рекурсивный запуск описанной функции (третий случай), и обе ветви рекурсии не завершились на следующем шаге. Это означает, что полуинтервал запроса $[a, b)$ покрывает

середину, то есть точку разбиения полуинтервала, соответствующего текущей вершине.

Рассмотрим теперь ветвь рекурсивных запусков, соответствующих левому поддереву. Так как правый конец полуинтервала всегда покрыт полуинтервалом запроса, может иметь место один из двух случаев:

- левое поддерево полностью снаружи запроса, правое покрыто частично;
- левое поддерево покрыто частично, правое покрыто полностью.

Оба эти варианта оставляют одну ветвь, не завершающуюся за один переход, то есть выполнят не более логарифмического количества рекурсивных вызовов.

Правое поддерево аналогично произведёт логарифмическое количество вызовов, домножив таким образом ответ на два.

Данное доказательство, в частности, показывает, что любой полуинтервал можно представить в виде объединения логарифмического числа непересекающихся полуинтервалов, соответствующих вершинам дерева.

1.6 Детали реализации

Выделим общие детали в реализации обхода дерева.

- Разбирается случай отсутствия пересечения: он наиболее нейтрален и не должен влиять на состояние.
- Разбирается случай полного покрытия. Он должен также разбираться за константное время.
- Остальные случаи не разбираются отдельно, однако нужно уметь комбинировать ответ именно в этот момент. Это единственный случай взаимодействия с детьми вершины.

1.7 Групповые операции

1.7.1 Присваивание на отрезке

Научимся выполнять операцию приваивания значения на отрезке: всем элементам a_i $(l \le i < r)$ присвоить значение x.

Будем выполнять обход дерева, как и в случае с запросом получение значения на полуинтервале.

Случай отсутствия пересечения не требует вмешательства.

В случае полного покрытия нужно полностью перезаписать все значения в поддереве. Так как реализовать эту операцию в реальном времени с сохранением времени работы возможности нет, запишем необходимость этого изменения в текущей вершине и оставим эту операцию отложенной.

Теперь в случае рекурсивного перехода к детям нужно учесть существующую отложенную операцию.

1.7.2 Проталкивание вниз

С появлением в дереве отложенных операций теперь правильно научиться их обрабатывать в общем случае. Заметим, что необходимую отложенную операцию нужно выполнять во всех случаях обработки вершины, требующих перехода к её детям.

Сама операция освобождения вершины v от отложенной операции очень проста: она применяет в аналогичном виде операцию к детям v_l и v_r вершины v, а затем пересчитывает значение искомой функции v_{total} для вершины.

Таким образом, общий алгоритм обхода дерева выглядит так:

1.7.3 Операция запроса на отрезке

Реализация незначительно дополнена относительно версии без групповых операций.

Пусть нужно вычислить ответ на полуинтервале $[a, b)$. Будем выполнять следующую рекурсивную функцию:

- Пусть текущая вершина дерева v соответствует полуинтервалу $[l, r)$
- Выполним операцию проталкивания вниз из текущей вершины. После этого в ней корректно посчитано значение искомой функции на $[l, r)$, а счетчики отложенных операций (прибавления, присваивания на отрезке) обнулены.
- Если $[a, b)$ не пересекается с $[l, r)$, полуинтервал $[l, r)$ учитывать не нужно — нужно вернуть нейтральный элемент относительно операции.
- Если $[a, b)$ полностью покрывает $[l, r)$, полуинтервал $[l, r)$ нужно учесть полностью — учтём его, выполнив операцию со значением, хранящемся в вершине v, после чего завершим функцию
- Во всех остальных случаях запустим функцию рекурсивно от детей v и выполним операцию для полученных значений.