Дерево отрезков. Построение

Дерево отрезков (англ. *Segment tree*) — это структура данных, которая позволяет за асимптотику $O(\log n)$ реализовать любые операции, определяемые на множестве, на котором данная операция ассоциативна, и существует нейтральный элемент относительно этой операции, то есть на моноиде. Например, суммирование на множестве натуральных чисел, поиск минимума на любом числовом множестве, перемножение матриц на множестве матриц размера $N * N$, объединение множеств, поиск наибольшего общего делителя на множестве целых чисел и многочленов.

При этом дополнительно возможно изменение элементов массива: как изменение значения одного элемента, так и изменение элементов на целом подотрезке массива, например разрешается присвоить всем элементам $a[l \dots r]$ какое-либо значение, либо прибавить ко всем элементам массива какое-либо число. Структура занимает $O(n)$ памяти, а ее построение требует $O(n)$ времени.

Содержание

Структура

Структура представляет собой дерево, листьями которого являются элементы исходного массива. Другие вершины этого дерева имеют по 2 ребенка и содержат результат операции от своих детей (например минимум или сумму). Таким образом, корень содержит результат искомой функции от всего массива $[0 \dots n-1]$, левый ребёнок корня содержит результат функции на $[0 \dots \frac{n}{2}]$, а правый, соответственно результат на $[\frac{n}{2}+1 \dots n-1]$. И так далее, продвигаясь вглубь дерева.

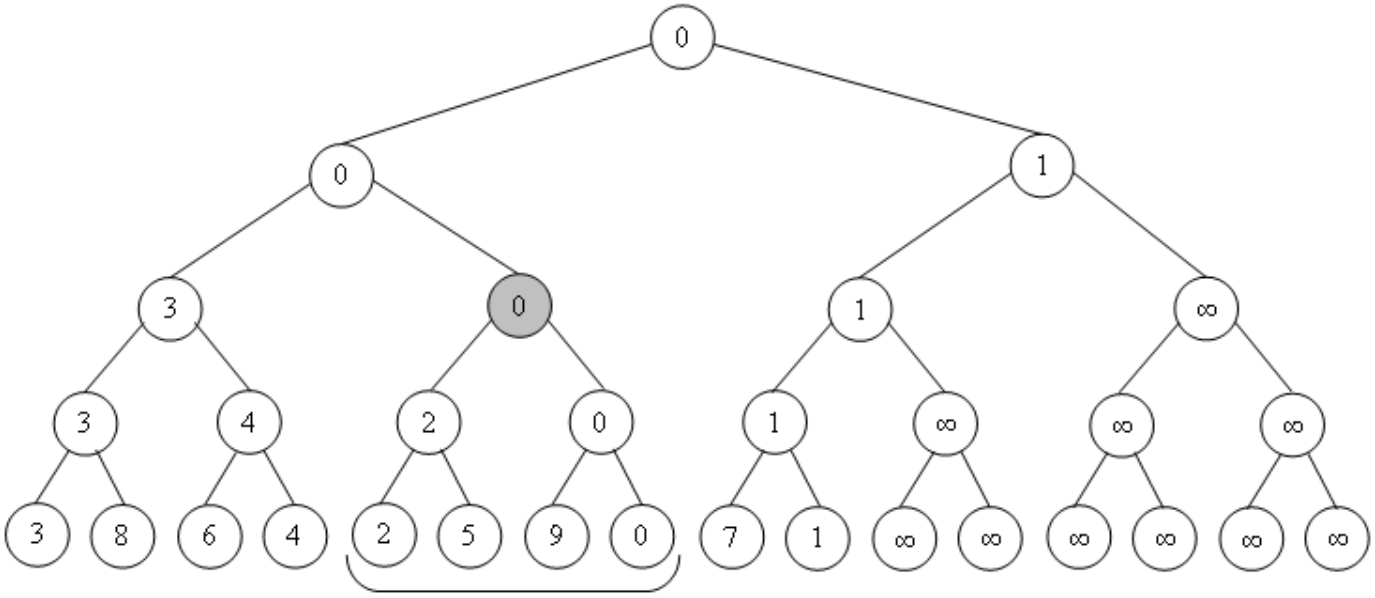
Построение дерева

Пусть исходный массив a состоит из n элементов. Для удобства построения увеличим длину массива a так, чтобы она равнялась ближайшей степени двойки, т.е. 2^k, где $2^k \geqslant n$. Это сделано, для того чтобы не допустить обращение к несуществующим элементам массива при дальнейшем процессе построения. Пустые элементы необходимо заполнить нейтральными элементами моноида. Тогда для хранения дерева отрезков понадобится массив t из 2^{k+1} элементов, поскольку в худшем случае количество вершин в дереве можно оценить суммой $n + \frac{n}{2} + \frac{n}{4} \dots + 1 < 2n$, где $n = 2^k$. Таким образом, структура занимает линейную память.

Процесс построения дерева заключается в заполнении массива t. Заполним этот массив таким образом, чтобы i-й элемент являлся бы результатом некоторой бинарной операции (для каждой конкретной задачи своей) от элементов с номерами $2i+1$ и $2i+2$, то есть родитель являлся результатом бинарной операции от своих сыновей (обозначим в коде эту операцию как "$\circ$"). Один из вариантов — делать рекурсивно. Пусть у нас имеются исходный массив a, а также переменные `tl` и `tr`, обозначающие границы текущего полуинтервала. Запускаем процедуру построения от корня дерева отрезков ($i = 0, \texttt{tl} = 0, \texttt{tr} = n$), а сама процедура построения, если её вызвали не

от листа, вызывает себя от каждого из двух сыновей и суммирует вычисленные значения, а если её вызвали от листа — то просто записывает в себя значение этого элемента массива (Для этого у нас есть исходный массив a). Асимптотика построения дерева отрезков составит, таким образом, $O(n)$.

Выделяют два основных способа построения дерева отрезков: построение снизу и построение сверху. При построении снизу алгоритм поднимается от листьев к корню (Просто начинаем заполнять элементы массива t от большего индекса к меньшему, таким образом при заполнении элемента i его дети $2i + 1$ и $2i + 2$ уже будут заполнены, и мы с легкостью посчитаем бинарную операцию от них), а при построении сверху спускается от корня к листьям. Особенные изменения появляются в реализации запросов к таким деревьям отрезков.



Реализация построения сверху:

```
function treeBuild(T a[], int i, int tl, int tr):  // мы находимся в вершине с номером i, который отвечает за полуинтерв
ал [tl, tr)
    if tl == tr
        return
    if tr - tl == 1
        t[i] = a[tl]
    else
        tm = (tl + tr) / 2                          // середина отрезка
        treeBuild(a, 2 * i + 1, tl, tm)
        treeBuild(a, 2 * i + 2, tm, tr)
        t[i] = t[2 * i + 1] ∘ t[2 * i + 2]
```

Реализация построения снизу:

```
function treeBuild(T a[]):
    for i = 0 to n - 1
        t[n - 1 + i] = a[i]
    for i = n - 2 downto 0
        t[i] = t[2 * i + 1] ∘ t[2 * i + 2]
```

См. также

- Реализация запроса в дереве отрезков сверху
- Реализация запроса в дереве отрезков снизу

- Несогласованные поддеревья. Реализация массового обновления

Источники информации

- Хабрахабр — Статья Максима Ахмедова (http://habrahabr.ru/post/115026/)
- Дискретная математика: Алгоритмы — Визуализатор дерева отрезков (http://rain.ifmo.ru/cat/view.php/vis/trees/segment-2006)
- MAXimal :: algo :: Дерево отрезков (http://e-maxx.ru/algo/segment_tree)
- Википедия — Дерево отрезков (http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BE%D1%82%D1%80%D0%B5%D0%B7%D0%BA%D0%BE%D0%B2)
- Википедия — Моноид (http://ru.wikipedia.org/wiki/Моноид)

- Эта страница последний раз была отредактирована 24 декабря 2018 в 20:59.