Танго-деревья

Шевкунов Семён

Аннотация

Танго-деревья представляют собой онлайновые бинарные деревья поиска (BST), которые достигают конкурентноспособности $O(\log \log n)$ в сравнении с оптимальной оффлайновой реализацией. На настоящий момент это самая оптимальная (среди доказанных) реализация BST.

1 Перед началом

Определение 1 (Бинарное дерево поиска)**.** Бинарное дерево поиска (BST) — *структура для работы с упорядоченным множеством* ключей $K = \{k_1, \ldots, k_n\}$*. Для любого узла с ключом* $k \in K$ *верно, что в узлах его левого поддерева содержатся ключи меньше* k*, а в узлах правого — больше* k*.*

Наивная реализация BST реализует операции вставки, удаления и поиска элемента за $O(\log n)$, где $n = |K|$, в среднем случае. В худшем случае асимптотика этих операций обращается в $O(n)$.

Красно-черное дерево — пример самобалансирующегося BST, которое обеспечивает сложность $O(\log n)$ в любом случае. Сбалансированность достигается за счет идеи хранить для каждого узла его «цвет»: «красный» или «черный». Эта структура нативно поддерживает за логарифмическую стоимость операции *разбиения* (split) и *конкатенации* (concatenate).
Определение 2. *Разбиение* Разбиением *красно-черного дерева* T *по узлу* x *назовем красно-черное дерево* T' *с корнем в узле* x*, хранящие ключи из дерева* T*.*

Ясно, что в описанном разбиении левое поддерево x хранит все ключи T, меньшие x, а правое — больше x.
Определение 3. *Конкатенация* Конкатенацией *красно-черных деревьев* P *и* T *по общему корню* x *назовем красно-черное дерево* T' *с корнем в* x*, где в левом поддереве содержатся все ключи из объединения ключей* P *и* T*, меньшие* x*, а в правом — большие* x*.*

Подробное описание работы красно-черного дерева выходит за рамки темы; в дальнейшем нам пригодится лишь факт, что его высота растет логарифмически от числа узлов.

2 Устройство танго-дерева

Опорное дерево[‡]

Работая с танго-деревом мы будем симулировать работу традиционного BST — *опорного дерева* (reference tree). В непосредственной реализации структуры опорное дерево не возникает явно — это абстракция, которая стоит за элементами танго-дерева, которые будут далее рассмотрены.

Факт 1. *Высота опорного дерева есть* $O(\log n)$.

Предпочтительные пути

Определение 4 (Предпочтительный сын)**.** *Рассмотрим в опорном дереве поддерево* T *с корнем в узле* n *с дочерними узлами* ℓ *и* r *— левым и правым, соответственно. Назовем* r предпочтительным *сыном (preferred child)* n*, если последний посещенный узел поддерева* T *находится в поддереве с корнем в* r*. Иначе предпочтительным сыном назовем* ℓ*.*

Определение 5 (Предпочтительное ребро)**.** *Если* m *— предпочтительный сын* n*, назовем* предпочтительным *ребро (preferred edge), инцидентное* m *и* n*.*

Определение 6 (Предпочтительный путь)**.** Предпочтительным путем *(preferred path) назовем максимальную по включению последовательность предпочтительных ребер в опорном дереве.*

Вспомогательные деревья

Узлы предпочтительных путей будем хранить во *вспомогательных деревьях* (auxiliary tree), которые по сути являются красно-черными деревьями. Для любого узла p из некоторого предпочтительного пути существует непредпочтительный сын c, который является корнем очередного вспомогательного дерева (заметим, что c может перестать быть корнем после балансировки вспомогательного дерева).

Соединим опорные деревья в итоговую структуру следующим образом. Для всякого ранее описанного узла p создадим ссылку на корень дерева, содержащий его непредпочтительного сына c. Для различения дочернего вспомогательного дерева и дочернего узла вспомогательного дерева пометим особенным образом корень очередного дерева. Полученное дерево деревьев назовем *танго-деревом*. В корне танго-дерева лежит вспомогательное дерево, содержащее вешины предпочтительного пути из корня опорного дерева.

3 Алгоритмическая составляющая

Поиск

Поиск ключа x в танго-дереве логически повторяет поиск в опорном BST. Отличие состоит в том, что путь поиска разбит на фрагменты, каждый из которых лежит целиком внутри одного предпочтительного пути и, следовательно, внутри одного вспомогательного дерева. Внутри такого дерева поиск осуществляется как в обычном красно-черном дереве. Если при сравнении ключей становится ясно, что дальнейший переход в опорном дереве должен происходить по непредпочтительному ребру, поиск во вспомогательном дереве

[‡]Эстетика терминологии танго-деревьев в (немногочисленных) русскоязычных источниках оставляет желать лучшего (чего только стоят «жирные ребра»). Здесь и далее используется авторский перевод английских терминов с указанием оригинала.

останавливается, и алгоритм переходит по сохраненной ссылке в другое вспомогательное дерево. Процесс продолжается до тех пор, пока ключ не будет найден или пока поиск не упрется в пустое поддерево.

Перестройка танго-дерева

Для сохранения инварианта, данного в определении предпочтительного ребра о том, что оно связывает родителя с последним посещенным сыном, нам необходимо после каждого запроса на поиск обновлять структуру танго-дерева. Рассмотрим для этого наше дерево вспомогательных деревьев.

Помимо ключа, для каждого узла вспомогательного дерева сохраним дополнительную информацию. Во-первых, будем хранить в каждом узле его глубину в опорном дереве Тогда потребуем следующие опреации, возможные над вспомогательными деревьями:

- *Поиск* элемента по ключу x в дереве (нативно реализовано для красно-черного дерева).
- *Разрез* (cut) — резделение единого вспомогательного дерева на два — одно хранит предпочтительный путь до заданной глубины d, второе — часть предпочтительного пути после d.
- *Склейка* (join) — объединение в одно дерево двух различных деревьев, хранящих последовательные непересекающиеся части предпочтительного пути.

Требуемая сложность операций — $O(\log k)$, где k — число узлов деревьев, над которами производится операция.

Во-вторых, сохраним для каждого узла минимальную и максимальную глубину, которая возможна среди всех узлов его поддерева. Понятно, что сохраненная нами дополнительная информация влияет на конечную сложность структуры не более чем константно.

Покажем, как реализовать операции разреза и склейки при помощи операций разбиения и конкатенации.

Рассмотрим разрез вспомогательного дерева A по глубине d. Заметим, что в A возможно найти минимальный ключ с глубиной больше, чем d, за $O(\log k)$. Используя запомненную максимальную глубину на поддереве найдем искомый минимальный узел ℓ, спускаясь влево от узлов, чья максимальная глубина на поддереве больше d. Аналогично найдем максимальный узел r на глубине больше, чем d. Для узлов ℓ и r также запомним их предцессор и сукцессор ℓ' и r', соответственно (посик предцессора и сукцессора — стандартная операция на BST).

Разобьем A на B и C по ℓ'. Так как в C содержатся ключи, большие ℓ' и ключ r' больше ключа ℓ', узел r' содержится в C. Разобьем C по r' на деревья D и E. Пометим крень D как корень нового вспомогательного дерева, как было описано ранее.

Заметим, что теперь D содержит исключительно ключи интервала (ℓ', r'), что соответствует отрезку $[\ell, r]$ — все узлы, которые глубже d. Нам осталось только конкатенировать по r' его правое поддерево E с левым (которое после отрезания D стало пустым) в исходное дерево C. Затем получим исходное дерево A конкатенацией B и C по общему корню ℓ', в итоге получая дерево A с отрезанным деревом D, в котором все узлы находятся в опорном дереве глубже, чем d.

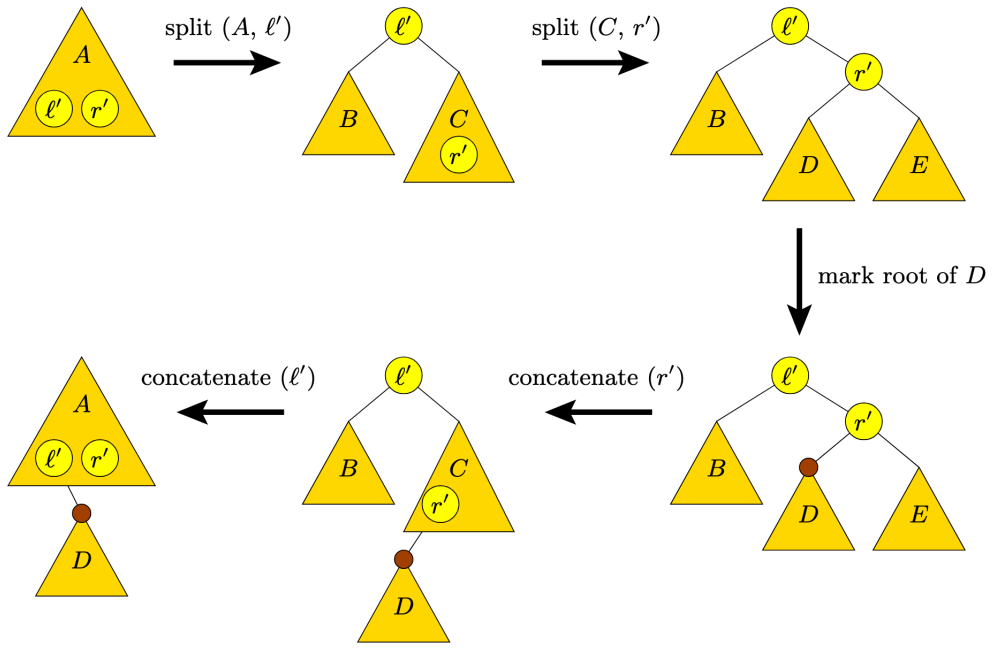

Рис. 1: Операция разреза дерева A по глубине d.

Рассмотрим склейку двух вспомогательных деревьев A и B. Сперва поймем, какое из деревьев граним нижнюю часть предпочтительного пути, сравнив их глубины в корнях (далее для простоты допустим, что в A хранятся узлы глубже, чем в B). Как и в случае разрезов, ключи в B находятся в интервале между некими узлами ℓ' и r' из A. Найдем в A помеченный корень B. Разобьем A по ℓ' и r', тогда корень B есть левый сын r'. Уберем метку с корня B и произведем конкатенацию по r' и l'. Таким образом имеем дерево для «склеенного» предпочтительного пути из путей деревьев A и B.

Применим описанные операции склейки и разреза для поддержания предпочтительных путей в танго-дереве. При обходе во время поиска ключа x_i, достигнув вершины x на глубине d, мы режем предпочтительный путь, содержащий предка x, по глубине $d-1$, и склеиваем верхнюю часть предпочтительного пути с предпочтительным путем, имеющим корень в x. Таким образом, предпочтительное ребро из родителя всегда будет вести к последнему посещенному сыну.

4 Анализ асимптотики

Лемма 1. *Поиск по ключу x_i производится за $O((k+1)(1+\log\log n))$, где k есть количество узлов, чьей предпочтительный сын сменился за время прохождения по дереву.*

Доказательство. Очевидно, что k сменивших предпочтительного сына узлов разделяют $k+1$ пройденных предпочтительных подпутей, которым соответствуют $k+1$ вспомогательных деревьев в танго-дереве. Так как высота опорного дерева есть $O(\log n)$, высота каждого вспомогательного дерева есть $O(\log\log n)$, и соответственно поиск по каждому опорному дереву есть $O(\log\log n)$. Отсюда, сложность поиска есть $O((k+1)(1+\log\log n))$.

Заметим, что перестройка танго-дерева имеет ту же сложность. Так как операции разреза и склейки суть последовательности фиксированного количества операций разбиения и конкатенации на красно-черных деревьях, время их выполнения идентично в точно-

сти до константы. Из-за того, что операции конкатенации и разбиения выполняются за $O(\log\log n)$, и смена предпочтительного сына происходит $k+1$ раз, имеем ту же сложность $O((k+1)(1+\log\log n))$. разреза □

Определение 7. *Interleave bound Пусть дана последовательность ключей на поиск* $X_m = x_1, \ldots, x_m$. *Определим* $\mathrm{IB}(X_m)$ *как количество операций (смен предпочтительных сынов) за время последовательного выполнения всех запросов из* X_m.

Теорема 1. *Время работы танго-дерева на последовательности ключей-запросов* X *длины* m *из множества мощностью* n *есть* $O((\mathrm{OPT}(X)+n)(1+\log\log n))$ *, где* $\mathrm{OPT}(X)$ *есть стоимомть оптимальноого оффлайнового BST на* X.

Доказательство. Так как общее количество смен предпочтительных сыновей есть $\mathrm{IB}(X)$ и всего максимум n узлов могут быть в первый раз назначены предпочтительными, имеем общее количество изменений на предпочтительных сыновьях $\mathrm{IB}(X)+n$. Основываясь на лемме, стоимость танго дерева тогда есть $O((\mathrm{IB}(X)+n+m)(1+\log\log n))$. Можно показать, что $\mathrm{OPT}(X) \geqslant \dfrac{\mathrm{IB}(X)}{2} - n$, причем также $\mathrm{OPT}(X) \geqslant m$. Значит сложность танго-дерева есть $O((\mathrm{OPT}(X)+n)(1+\log\log n))$. □

Если m сильно превышает n, имеем сложность $O((\mathrm{OPT}(X))(1+\log\log n))$, что дает конкурентоспособность $O(log\log n)$ в сравнении с оптимальным оффлайновым BST.

Источники

- E. Demaine & al.: Dynamic Optimality — Almost
- https://en.wikipedia.org/wiki/Tango_tree