Статически оптимальное дерево отрезков

Артем Комендантян, 4 курс МФТИ, кафедра анализа данных
Научный руководитель: Михаил Тихомиров, к.ф.-м.н.

2021

Определения

Пусть дан массив из n элементов

Дерево отрезков - это двоичное дерево, в котором есть

- n листьев, соответствующих отрезкам единичной длины
- Вершины с двумя сыновьями. Правый сын соответствует отрезку, следующему сразу за отрезком левого сына. Вершина соответствует объединению отрезков сыновей

Корень дерева соответствует всему массиву (отрезку $[1; n]$)

Определения

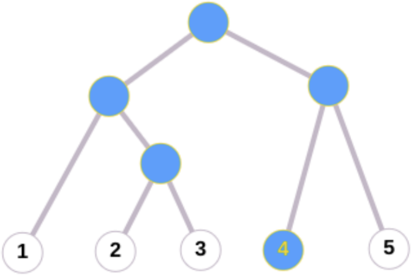
Запрос сверху на отрезке $[L, R]$ начинается в корне.

Если сейчас рассматривается вершина, отрезок которой не лежит полностью в отрезке $[L, R]$, то запрос рекурсивно вызывается от тех сыновей, отрезки которых пересекаются с $[L, R]$. Иначе рекурсивных вызовов от сыновей не происходит.

В обоих случаях вершина считается посещенной и в ней выполняются какие-то действия, специфичные для запроса.

Пример

Дерево отрезков на пяти элементах
Запрос на отрезке $[2; 4]$
Посещено пять выделенных вершин



Постановка задачи

Дано: Распределение вероятностей на запросах-отрезках с границами из $[1; n]$

Необходимо: построить дерево отрезков, для которого минимально среднее количество посещенных вершин при запросах сверху.

Интересует как точное решение за как можно более лучшую асимптотику, так и приближенное за сложность нахождения $O(n+S)$ или $O(n+S\log S)$, где S - количество отрезков с ненулевой вероятностью.

Мотивация

- Дерево отрезков - мощная структура, позволяющая эффективно решать множество задач. Примеры: наименьший общий предок двух вершин в дереве, площадь объединения прямоугольников со сторонами, параллельными осям координат
- Потенциальное ускорение в реальных задачах
- Аналогичная задача для деревьев поиска хорошо изучена

Статически оптимальное двоичное дерево поиска

Дано: множество из n упорядоченных элементов и $2n+1$ вероятностей, что запрос будет равен элементу из множества, либо находиться между двух соседних элементов

Необходимо: построить двоичное дерево поиска, для которого минимально среднее количество посещенных вершин при запросах

Обзор решений

- **Точное решение:** динамическое программирование за $O(n^3)$ времени и $O(n^2)$ памяти, оптимизируется до $O(n^2)$ времени [1].
- Обобщение задач, к которым применима данная оптимизация [3]
- Точное решение за $O(n \log n)$ времени для случая, когда запросы поступают только между элементов [4]
- **Приближенное решение:** жадное решение за O(n), хуже оптимального не более чем в константу раз, где константа примерно равна 2.283... [2]
  Идея решения - выбирать такой корень, чтобы максимально уравнять сумму вероятностей в левом и правом поддереве, затем перейти к двум меньшим подзадачам

Вернемся к нашей задаче

Точное решение

- Нам даны веса-вероятности запросов
- $dp_{L,R}$ - суммарный вес оптимального дерева отрезков, построенного на точках с L по R
- Для каждого запроса оставляем пересечение с $[L, R]$
- Либо не рассматриваем, если не пересекается

Точное решение

- Для отрезка $[L, R]$ к весу дерева отрезков прибавляется сумма весов запросов, которые пересекаются с $[L, R]$
- Если сыновьям корня соответствуют отрезки $[L, m]$ и $[m+1, R]$, надо учесть вес дерева отрезков на $[L, m]$ и на $[m+1, R]$
- Запросы целиком содержащие $[L, R]$ не нужно учитывать в сыновьях

Точное решение

- Отсюда $\mathrm{dp}_{L,R} = cost_{L,R} + \min\limits_{L \le m < R} dp_{L,m} + dp_{m+1,R}$
- Восстановление дерева по оптимальным m
- Функция стоимости $cost_{L,R}$ ищется перебором за $O(n^2 S)$
- Можно выразить её слагаемые рекурсивно и находить за $O(n^2)$
- Итоговое решение за $O(n^3)$ времени и $O(n^2)$ памяти

Идеи для более оптимального решения

В [3] было показана возможность оптимизации до $O(n^2)$, если

- $\forall a \leq b \leq c \leq d: \ cost_{a,d} \geq cost_{b,c}$ (монотонность)
- $\forall a \leq b \leq c \leq d: \ cost_{a,c} + cost_{b,d} \leq cost_{a,d} + cost_{b,c}$ (quadrangle неравенство)

В нашей задаче

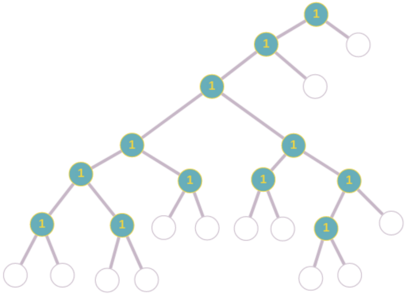
- Первое неравенство верно
- Второе верно с противоположным знаком

Приближенное решение

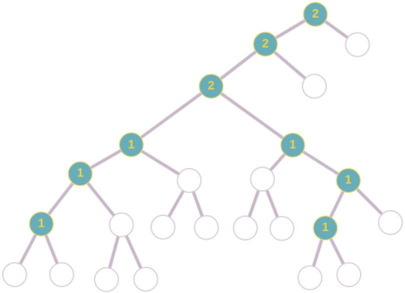
- Сейчас: Число посещенных вершин запроса $[L, R]$
- Меняем на сумму максимальных глубин запросов $[1, R]$ и $[L, n]$
- Веса запросов оставляем те же
- Было доказано, что старое и новое значение запросов отличаются не более чем в два раза
- Оптимальное решение для новой задачи не более чем в четыре раза хуже оптимального для старого

Частный случай запроса

Посещенные вершины $[L, R]$



Сумма глубин $[1, R]$ и $[L, n]$



(для каждой вершины отмечено, сколько она раз учтётся в ответе)

- До какой-то глубины вклад в ответ в два раза больше справа
- После этого не более чем в два раза больше слева
- Общий случай разбирается похожим образом

Приближенное решение

- lw_i - сумма весов запросов $[i, n]$
- rw_i - сумма весов запросов $[1, i]$
- $w_i = rw_i + lw_{i+1}$
- $dp_{L,R}$ - стоимость оптимального дерева отрезков с новыми запросами, построенного на точках с L по R
- $dp_{L,R} = \sum\limits_{L}^{R-1} w_i + \min\limits_{L \le m < R} dp_{L,m} + dp_{m+1,R}$
- Эквивалентно задаче об оптимальном дереве поиска с вероятностями элементов пропорциональными $w_1, \ldots, w_{n-1}$

Приближенное решение

Из сведения, [1] и [2] получаем два приближенных решения:

- За $O(n^2 + S)$ времени и не более чем в четыре раза хуже оптимального
- За $O(n + S)$ времени и не более чем в $\approx 4 \cdot 2.283 = 9.132$ раз хуже оптимального

Приближенное решение на практике

- Случайные тесты с $n \leq 100$ и S от 1 до n^2
- Решение за $O(n^2 + S)$ не более чем в ≈ 1.42 раз хуже оптимального
- Решение за $O(n + S)$ не более чем в ≈ 2.3 раз хуже оптимального
- Значения сравнимы с константой для приближенного нахождения дерева поиска
- И уже могут иметь не только теоретический интерес

Спасибо за внимание!

Список литературы

1. Knuth, Donald E. (1971), "Optimum binary search trees Acta Informatica.
2. Mehlhorn, Kurt (1975), "Nearly optimal binary search trees Acta Informatica.
3. F. Frances Yao (1980), "Efficient Dynamic Programming Using Quadrangle Inequalities".
4. Garsia, Adriano M.; Wachs, Michelle L. (1977), "A new algorithm for minimum cost binary trees SIAM Journal on Computing.