Первый курс, весенний семестр

Практика по алгоритмам #5

Euler Tour Trees, Heavy-Light Decompostion

Contents

1 Новые задачи

1. Дано дерево с весами на ребрах, надо эффективно обрабатывать запросы:
   a) $\min(a, b)$ — минимум на пути из вершины a в вершину b.
      $set(e, x)$ — присвоение ребру e веса x.
   b) $\min(a, b)$ — минимум на пути из вершины a в вершину b.
      $add(a, b, x)$ — прибавление числа x к весам всех ребер на пути из вершины a в вершину b.
2. Дано дерево с весами в вершинах. Веса из $\{0, 1\}$.
   a) Запросы: сделать вес вершины равным 1, найти ближайшого предка с весом 0.
   b) Запросы: сделать вес вершины равным 1, найти ближайшого предка с весом 1.
   c) Запросы: изменить вес вершины, найти ближайшего предка с заданным весом.
   d) Веса из $\mathbb{N}$. Запросы: присвоить вес вершины, по x найти ближайшего предка с весом $\geq x$.
3. Дерево с весами в вершинах. Нужно научиться отвечать на запрос $count(a, b, x)$ — количество вершин на пути из a в b, у которых вес $\geq x$, если:
   a) Веса не меняются. $\mathcal{O}(\log n)$.
   b) Теперь веса меняются. Запрос за $\mathcal{O}(\log^3 n)$, изменение веса за $\mathcal{O}(\log^2 n)$.
   c) (*) Веса меняются. Эйлеров обход. $\mathcal{O}(\log^2 n)$.
4. Придумайте модификацию Heavy-Light-Decomposition, поддерживающую добавлении новых листьев.
   a) $\mathcal{O}(\sqrt{n})$ на добавление листа, $\mathcal{O}(\sqrt{n} + \log^2 n)$ на запрос `get`.
   b) Добавление за $\mathcal{O}(\log^2 n)$. Подсказка: мы хотим быстро *split*-ить и *merge*-ить пути.
5. Дан лес с положительными целочисленными весами на ребрах. Обрабатывать запросы:
   a) $link(v, root, w)$ – подвесить корень одного дерева к вершине другого ребром веса w
   b) $cut(a, b)$ – удалить ребро
   c) $sum(a, b)$ – сумма весов рёбер на пути
   Все запросы нужно обрабатывать за $\mathcal{O}(\log n)$.
6. (*) Дано дерево вложенности изначально пустых namespace-ов. Чтобы, находясь в вершине дерева v, узнать значение переменной x, нужно подниматься по дереву, пока не попадём в вершину, где x определена. Все переменные в задаче булевы. Запросы:
   a) $set(v, x, value)$ — присвоить в пространстве имён v переменной с именем x значение $value$.
   b) $count(x)$ — посчитать, во скольких пространствах имен-листьях переменная с именем x имеет значение $true$, $false$ и во скольких не объявлена.

2 Домашнее задание

2.1 Обязательная часть

1. (4) Нужно модифицировать алгоритм для решения задачи LA за $\langle \mathcal{O}(n + k \log n), \mathcal{O}(1) \rangle$ (двоичные подъёмы только для k листьев + ladder longest path decomposition) до времени $\langle \mathcal{O}(n \log \log n), \mathcal{O}(1) \rangle$ без использования микро-макро эвристики (идеи четырёх русских). *Подсказка к одному из возможных решений: подняться от листьев на $\mathcal{O}(\log n)$ вверх.*
2. (4) Дано деерево с **фиксированным корнем**. Нужно построить статически оптимальное **покрытие дерева вертикальными путями** (каждая вершина лежит ровно в одном пути) для ответа на запросы "количество рёбер с весом не более X_i на пути дерева (a_i, b_i)". Запросы должны обрабатываться в online. Допустимая память на предподсчёт – $\mathcal{O}(n)$. Оптимальность в смысле суммарного количества прыжков между путями. "Статическая" – означает, что запросы даны в offline, а нам нужно построить оптимальную структуру данных, которая будет обрабатывать их в online.

2.2 Дополнительная часть

1. (4) Придумайте модификацию Euler Tour Trees для хранения леса подвешенных деревьев, поддерживающую операции Link, Cut, MakeRoot, IsAncestor(a, b).
2. (4) В Heavy-Light-Decomposition при подъёме вверх почти все запросы к внутреннему дереву отрезков считаются на префиксе. Изменение в точке делается за одно обращение к внутреннему дереву отрезков, то есть за $\mathcal{O}(\log n)$. Идея кеширования: сохраним в дереве отрезков для каждого префикса пару – когда последний раз считали, какую функцию получили. Если во время запроса "время последнего изменения дерева" меньше "времени, когда считали ответ для префикса", можно сразу вернуть уже готовый ответ. Запрос обновления: обновить в точке, поменять время изменения, работает за $\mathcal{O}(\log n)$. Оцените время на запрос `get`. Или приведите пример, на котором $\mathcal{O}(log^2 n)$, или докажите оценку $\mathcal{O}(\log n)$.
3. (8) Используя за основу Euler Tour trees, придумайте *online dynamic connectivity* за $\mathcal{O}(\log^2 n)$. Подсказка: при удаление ребра, если оно лежит в остове, нужно найти ему замену, при этом придётся перебрать рёбра-кандидаты, нужно перебрать поменьше рёбер-кандидатов и всем им уменьшить некий потенциал.