Домашнее задание №2
Алгоритмы. 5 курс. Весенний семестр.

Горбунов Егор Алексеевич

24 февраля 2016 г.

1 Мои решения

Задание №1 Дано дерево из одной вершины. Требуется уметь отвечать online за $\mathcal{O}(\log n)$ на запрос: подвесить вершину u к вершине дерева v и вернуть диаметр дерева. Диаметр дерева — длина самого длинного простого пути в дереве.

Решение: Будем для дерева T хранить концы максимального простого пути: x и y. Ясно, что такой путь не один, мы будем хранить для какого-то одного. Так же для вершины будем хранить её глубину в дереве, для того, чтобы уметь находить расстояние между 2-мя вершинами a и b через глубину a, b и $lca(a, b)$. Следующая процедура решает исходную задачу:

```python
def hang(u, v, T):
   u.parent = v
   u.depth = v.depth + 1
   dist_x_u = T.x.depth + u.depth - 2 * lca(T.x, u).depth
   dist_y_u = T.y.depth + u.depth - 2 * lca(T.y, u).depth
   if dist_x_u > T.diametr:
      T.y = u
   elif dist_y_u > T.diametr:
      T.x = u
   T.diametr = max(dist_x_u, dist_y_u, T.diametr)
   return T.diametr
```

Мы умеем искать $lca(a, b)$ за $\mathcal{O}(\log n)$, а значит и сложность данной процедуры $\mathcal{O}(\log n)$. Осталось показать её корректность. Для этого покажем, что если у нас есть некоторый максимальных путь $x...y$ в дереве T, то,в дереве $T + v$, где v— добавленный лист, максимальный путь будет $x...y$, $x...v$ или $y...v$. Действительно, если в $T + v$ диаметр не изменился в сравнении с T, то один из максимальных путей в $T + v$ — это $x...y$. Пускай теперь диаметр $T + v$ на 1 больше, чем в T (больше он измениться не мог, т.к. добавили лишь 1 вершину). Так же ясно, что диаметр мог увеличиться только засчёт того, что был продлён какой-то максимальный путь $a...b$. Любые 2 максимальных пути в дереве пересекаются, т.к. иначе можно было бы построить путь длиннее в силу связности дерева. Пересекаться 2 пути в дереве могут только по какому-то отрезку $c..d$, пусть он разбивает пути так (НУО): $x..c..d..y$, $va..c..d..b$. Ясно, что длина $x..c$ равна $a..c$ и длина $d..y$ равна длине $d..b$, иначе, выбирая максимумы из каждой пары можно было бы построить пути длиннее $x..y$ в T, а он уже максимальный. Таким образом путь $va..c..d..y$ — максимальный в $T+v$. Т.е. мы показали, что новый максимальный путь в $T+v$ стоит выбирать из: $x...y$, $x...v$ или $y...v$. А значит корректность показана. ■

Задание №2 Дан ориентированный граф, в котором исходящая степень каждой вершины равна единице. Запросы online: из вершины v сделать k шагов вперёд.

(a) Предподсчёт: $\mathcal{O}(n \log k_{max})$, время на запрос: $\mathcal{O}(\log k)$

(b) Предподсчёт: $\mathcal{O}(n \log n)$, время на запрос: $\mathcal{O}(\log \min(k, n))$

Решение:

(a) Нам дана длина максимального прыжка k_{max}, предподсчитаем двоичные прыжки для каждой вершины (т.к. из каждой вершины можно пойти в одну единственную) длин $0, 2^1, 2^2, \dots, 2^{\log k_{max}}$. Вершин всего n: $v_1, \dots, v_n$

```
for v ∈ {v_1, ..., v_n} do
    jump[v, 0] ← v
    jump[v, 1] ← v.next
for k ∈ 1 ... log k_max do
    for v ∈ {v_1, ..., v_n} do
        jump[v, 2^k] ← jump[jump[v, 2^{k-1}], 2^{k-1}]
```

Теперь будем отвечать на запрос сделать k шагов из вершины v так: прыгнем сначала в вершину $u = jump[v, 2^{\lfloor \log k \rfloor}]$, а потом пойдём вперёд от полученной вершины u. Ясно, что нам осталось сделать не более $\log k$ переходов. Таким образом отвечаем на запрос за $\mathcal{O}(\log k)$ с предподсчётом $\mathcal{O}(n \log k_{max})$.

(b) Не умаляя общности рассмотрим орграф, в котором лишь одна компонента слабой связности, т.е. слабо-связный орграф G в котором $outdeg(v) = 1$. Такой граф может содержать лишь один цикл, иначе была бы вершины с $outdeg(v) > 1$. Мы легко за $\mathcal{O}(n)$ сможем найти этот цикл. Пометим все вершины, которые принадлежат циклу, это тоже делается за $\mathcal{O}(n)$ и запомним длину этого цикла — len. Теперь, аналогично предыдущему пункту, предподсчитаем прыжки, но только для длин: $0, 2^1, 2^2, \dots, 2^{\log n}$. Теперь, если длина прыжка в запросе $\leqslant n$, то мы аналогично прыжками сможем получить ответ, используя предподсчитанные прыжки. Но если $k > n$, то заметим, что прыгнув на $n = 2^{\log n}$ мы точно окажемся на цикле (если он есть), т.к. всего вершин n. Таким образом, мы находимся на цикле длины len и нам осталось сделать $k - 2^{\log n}$ шагов. Но т.к. теперь мы будем шагать только по циклу, то имеет смысл прыгать на $(k - 2^{\log n}) \mod len$, а это число всяко $< n$. Таким образом нам хватило предподсчитанных прыжков $0, 2^1, 2^2, \dots, 2^{\log n}$, чтобы покрыть запросы для любых k. Предподсчёт $\mathcal{O}(n \log n)$, запрос $\mathcal{O}(\log \min(k, n))$.

Задание №3 Дан массив чисел длины n. За $\mathcal{O}(\log n)$ в online обрабатывать запросы:

- посчитать сумму кубов чисел на отрезке $[L, R]$
- прибавить x ко всем числам на отрезке $[L, R]$
- получить значение i-го числа

Решение: Пускай $s_3 = x_1^3 + x_2^3 + \ldots + x_k^3$, $s_2 = x_1^2 + x_2^2 + \ldots + x_k^2$ и $s = x_1 + x_2 + \ldots + x_k$,

$$(x_1 + a)^3 + (x_2 + a)^3 + \ldots + (x_k + a)^3 = x_1^3 + 3x_1^2 a + 3x_1 a^2 + a^3 + \ldots + x_k^3 + 3x_k^2 a + 3x_k a^2 = \\ = s_3 + 3a \cdot s_2 + 3a^2 \cdot s_1 + k \cdot a^3$$

Т.о. построим на данном массиве дерево отрезков, но теперь будем поддерживать одновременно сумму, сумму квадратов и сумму кубов на отрезках. Пересчёт суммы кубов на отрезке теперь просто пересчитывается: к каждому за $\mathcal{O}(\log n)$ отрезков, разбивающих $[L, R]$, применяем полученную выше формулу. Получение значение i-го элемента остаётся таким же: спускаясь по дереву добавляем к ответу значения, которые когда-либо прибавлялись к отрезку, содержащему это число. ■

Задание №4 Дана скобочная последовательность из круглых скобок длины n. Запросы: является ли отрезков $[L, R]$ правильной скобочной последовательностью; изменить i-ую скобку. $\mathcal{O}(\log n)$, online.

Решение: Рассмотрим следующий массив $P[0..n]$:

$$P[0] = 0$$
$$P[i] = \big[\text{число «(» скобок} - \text{число «)» скобок}\big] \text{ на отрезке } [1, i]$$

Знаем, что вся скобочная последовательность длины n является правильной, если $P[n] = 0$ и $\forall i \in [1..n]$ $(P[i] \geqslant 0)$, а иначе: $\min_{i \in [1..n]} P[i] \geq 0$. Аналогично можно обобщить: подпоследовательность $[L, R]$ исходной скобочной последовательности является правильной, если:

$$P[R] = P[L-1] \text{ и } \min_{i \in L..R} P[i] \geq P[L-1]$$

Вышенаписанное правило верно в силу того, что число откр. скобок минус число закр. скобок на любом префиксе $[L..i]$ подстроки $[L..R]$ равно $P[i] - P[L-1]$.

Построим над массивом $P[0..n]$ дерево отрезков. Мы умеем уже поддерживать операцию $\min$ на подотрезках, а значит умеем и отвечать на запрос о правильности скобочной подпоследовательности $[L..R]$ в силу описанного выше правила.

Заметим теперь, что изменение скобки на i-ой позиции исходной строки приводит к следующему:

$$\forall j \in [j..n]: \; P[j] \to P[j] - 2, \text{ если на } i\text{-ой позиции стояла «(»}$$
$$\forall j \in [j..n]: \; P[j] \to P[j] + 2, \text{ если на } i\text{-ой позиции стояла «)»}$$

Таким образом нам просто нужно уметь за $\mathcal{O}(\log n)$ выполнять прибавление числа на отрезке, но это мы уже умеем делать используя дерево отрезков.

Таким образом мы свели ответ на запрос о правильности скобочной подпоследовательности к поиску минимума на отрезке в массиве P, а запрос изменения скобки свели к прибавлению числа на отрезке в массиве P. Всё делаем за $\mathcal{O}(\log n)$. ∎

2 Дорешивание

Задание №5 Попробуем модифицировать идею `SparseTable` так, чтобы она работала для произвольных ассоциативных функций: предложите способ выделить $\mathcal{O}(n \log n)$ отрезков в массиве размера n так, что любой отрезок $[L, R]$ можно было представить в виде объединения $\mathcal{O}(1)$ непересекающихся выделенных отрезков. Заметим, что дерево отрезков выделяет $\mathcal{O}(n)$ орезков и любой отрезок представляется как объединение $\mathcal{O}(\log n)$ из них.

Решение: Рассмотрим дерево отрезков на массиве. И на любых двух сосендних отрезках на одном уровне посчитаем суффиксные суммы на левом отрезке и префиксные суммы на правом отрезке.

Задание №6 Дан массив из n элементов. Запросы: $k-e$ по порядку среди различных чисел на отрезке $[L, R]$.

(a) offline за $\mathcal{O}\left(\log^3 n\right)$

(b) online за $\mathcal{O}\left(\log^3 n\right)$

Решение:

(a) Задача про порядковую статистику и бинпоиск (?)

(b) (?)

3 Практика

Задание №1 Поставить на плоскость точку и за $\mathcal{O}\left(\log^2 n\right)$ отвечать на запрос, сколько точек в квадрате $[L..RxB..T]$.

Решение: hint: дерево отрезков, в вершине которого дерево отрезков.
Дерево поиска внутри дерева отрезков. Добавление — вставка с y-коордиантой в качестве ключа в соответствующие конкретному x деревья поиска. Таким образом у нас изначально пустая сетка $F[1..n][1..m]$. Строим дерево отрезков размера n (над массивом $F[1..n]$), а в каждом узле дерева храним пустое дерево поиска. Если нужно добавить точку с координатами

(i, j), то мы спусаемся по дереву отрезков к элементу i и в каждую посещённую вершину дерева отрезков, как в дерево поиска, добавляем j.
Запрос о числе точек в $[L..RxB..T]$ соответственно будет таким: дерево отрезков даёт на $\mathcal{O}(\log n)$отрезков-деревьев, на которые разбит отрезок $[L..R]$. В каждом таком дереве быстро ищем число вершин с ключами в заданном отрезке $[B..T]$. ■

Задание №2 Дано дерево с весами на рёбрах. Запрос online: минимум на пути из a в b за $\langle \mathcal{O}(n \log n), \mathcal{O}(\log n) \rangle$, т.е. $\mathcal{O}(n \log n)$ — предподсчёт и $\mathcal{O}(\log n)$ на запрос.

Решение: Предподсчитываем двоичные прыжки вместе с минимумом на прыжке, т.е. $jump[v, 2^k]$ и $min[v, 2^k]$, где последнее — это вес минимального ребра на пути вверх из v в $jump[v, 2^k]$ ■

Задание №3 Дерево с весами на рёбрах. Нужно отвечать online на запросы о длине простого пути между парой вершин. $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$

Решение: Рассматриваем эйлеров обход дерева, но каждое ребро вниз входит с плюсом (его вес), а каждое ребро вверх идёт с минусом. Далее на запрос о пути между a и b мы делаем запрос о сумме на отрезке между a и $lca(a, b)$ (по модулю) и складываем с суммой на отрезке между b и $lca(a, b)$.

Задание №4 Дерево, на вершинах которого могут быть пометки. Запросы: пометить вершину, снять пометку с вершины, число помеченных вершин в поддереве. $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$.

Решение: Эйлеров обход. Только нужно понимать, где границы поддерева, т.е. запоминать первое и последнее вхождения вершины.

Задание №8 Дан лес подвешенных деревьев. Нужно отвечать на запросы:

- Подвесить дерево с корнем v к вершине u другого дерева
- Отрезать поддерево с корнем в вершине v от её дерева
- Проверить, в одном ли дереве леэат u и v

Время: $\mathcal{O}(\log n)$.

Решение: Декартово дерево. Красим вершины в эйлеровом обходе. Вставляем перед белым (?) вхождением. Не ясно. ?????????????????????????????????

Задание №9 Дан лес подвешенных деревьев. Нужно отвечать на запросы:

- `LCA` вершин u и v

Время: $\mathcal{O}(\log n)$.

Решение: Та же структура, что и в 8? Белые вершины за $+1$, чёрные за -1 и минимум на префиксе.