The Summer School Programming
Uzhgorod 2020
Лига I - День 5

Тема дня: Алгоритмы на бинарных деревьях

Морозов Илья, Камеко Валерий, Парамонов Антон

Разбор задач

Задача А. Репликация – часть 1

Давалось N запросов вида:

- '$+\ A_i$' – добавить элемент в структуру $(A_i < A_{i+1})$.
- '$-$' – удалить из структуры минимальный элемент.

После каждого запроса требовалось вывести количество обменов, а также минимальный элемент.

Задача A. Репликация – часть 1. Решение $O(NlogN)$

«Генное дерево» — это эквивалент структуры данных «бинарная куча». Будем отвечать на запросы:

- На добавление – достаточно лишь добавить элемент в конец кучи, ‘замен’ не будет верхний элемент не изменится.
- На удаление – в функции удаления будем поддерживать счетчик обменов по мере спуска корневого элемента.

В худшем случае мы потратим на запрос второго типа $O(logN)$ времени, на запрос первого типа $O(1)$.

Итого время на запрос – по прежнему $O(logN)$.

Задача B. Репликация – часть 2

Давалось N запросов вида:

- '$+ A_i$' – добавить элемент в структуру.
- '$-$' – удалить из структуры минимальный элемент.

После каждого запроса требовалось вывести количество обменов, а также минимальный элемент.

Задача B. Репликация – часть 2. Решение $O(NlogN)$

Будем решать задачу как 'Задача A. Репликация – часть 1'.

Однако дополнительно требуется реализовать операцию добавления в кучу с поддержанием количества обмена в ее процессе.

Время на запрос – по прежнему $O(logN)$.

Задача C. Заражение.

Было дерево, состоящее из $E-1$ вершин с весами C_i.

Нужно было распределить N вирусов с временами жизни A_i на 'конечные поверхности', если все они начинают с поверхности под номером S.

При этом требовалось максимизировать суммарное остаточное время жизни:

$$T = \sum_{i=1}^{E-1} A_i - C_{S..V_i}$$

Где V_i – конечная поверхность к которой пойдет вирус i.

Задача C. Заражение. Решение $O(NlogN) + O(ElogE)$

'Конечные поверхности' – листья в дереве.

Найдем все листья достижимые из вершины S и отсортируем их по возрастанию расстояния от вершины S ($C_{S..V_i}$).

Далее отсортируем все вирусы по убыванию их времени жизни (A_i).

Сопоставим каждому вирусу в отсортированном списке — лист в другом списке.

Тогда ответом, для выбранного i-го вируса будет являтся $max(A_i - C_{S..V_i}, 0)$. Ответом является сумма этих значений.

Задача D. Коэволюция.

Давалось две последовательности $A_{1..N}$ и $B_{1..M}$.

Нужно было найти сумму индексов перемещений, выполняя одну операцию пока это возможно.

Операция состояла в том чтобы взять начала из двух последовательностей удалить их и добавить их в ту последовательность в которой начало было больше*, так чтобы соблюдалось свойство неубывания.

При этом к ответу добавляется индекс элемента после вставки.

Задача D. Коэволюция. Решение $O((N+M)log(N+M))$

Используя одну из структур данных которая позволяет работать с последовательностями (Treap, Splay Tree, ...), а именно поддерживают:

- Вставку в нужную позицию, с возвратом индекса.
- Удаление из начала.

Для этого подходит например Treap по неявному ключу:

- Вставку – можно сделать разрезав дерево по нужному индексу, а потом все склеить с элементом.
- Удаление – производиться разрезами.

Задача D. Коэволюция. Решение $O((N+M)log(N+M))$

Можно воспользоваться уже готовым решением...

```
...
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

using namespace __gnu_pbds;

typedef tree<
    long long,
    null_type,
    std::less_equal<long long>,
    rb_tree_tag,
    tree_order_statistics_node_update> indexed_multiset;
...
answer += set_1.order_of_key(index);
...
```

Задача E. Синтез и борьба.

Было дано дерево из N вершин.

Требовалось выбрать множество центров, а также назначить оставшимся вершинам центры, так чтобы минимизировать суммарную стоимость затрат.

При этом:

- Построить центр стоит K.
- Присоединить вершину v к центру стоит d_{len} – для len – длины от центра до v.

Задача E. Синтез и борьба. Решение $O(N^2)$

Рассмотрим некоторый план выбора центров.

Тогда очевидно, что для каждого оставшегося города выгодно взять ближайший центр ($D_i <= D_j$, для $i < j$). Значит – вершины вместе с назначенными к ним вершинами представляют собой поддеревья исходного дерева.

Задача E. Синтез и борьба. Решение $O(N^2)$

Для решения задачи будем поддерживать несколько массивов для подсчета ДП:

- $R_{v,u}$ — расстояние от u до v.
- $F_{v,g}$ — ответ на задачу для всего поддерева v, если g является одним из центров.
- P_v – такой центр, для которого F_{v,P_v} — максимальное

Для подсчета ДП, нужны переходы:

- $F_{v,j} = K + D_{R_{v,j}}$ — для центра в вершине j + стоимость для вершины v
- $F_{v,j} += min(F_{u,j} - K, F_{u,P_u})$ — либо мы добавляем вершину u (соседнюю с v) к текущему центру j, либо отдаем ее в к некоторому другому центру в вершине P_u.

Ответом является $F_{root,P_{root}}$.

Задача F. Интересный белок.

Давалось дерево из N вершин с весами вершин w_i.

Требовалось ответить на Q запросов u_j v_j:

- какую максимальную сумму весов на пути u_j - v_j можно получить с помощью одного 'поворота' пути a - b, если ни a, ни b не должны лежать на пути u_j - v_j.

Задача F. Интересный белок. Решение $O(QlogN)$

Постоим новый граф G' – в котором вершины это циклы, а ребра обозначают была ли связь между вершинами двух циклов.

Рассмотрим запрос u v – на сжатом графе u' v'. Если на пути между u' и v' есть хотя бы один цикл размера > 3, то добавление ребра будет нарушать 'свойство'.

Для определения того есть ли на пути цикл, можно воспользоваться DP с LCA.

Задача F. Интересный белок. Решение $O(QlogN)$

Посчитаем ДП:

$$F_{v'} = F_{p'} + C_{v'}$$

Где $C_{v'}$ – 1 если вершина является циклом с более чем 3 вершинами, иначе 0.

Для ответа на запрос достаточно посчитать $F_{u'} + F_{v'} - 2 * F_{lca(u',v')}$.

Если значение > 0, то ответ на запрос 'No', иначе 'Yes'.

Задача F. Интересный белок. Решение $O(Q + N)$

Т.к. запросы поступают оффлайн, то можно улучшить решение до $O(Q + N)$ с помощью алгоритма Тарьяна.

Задача G. Восстановление паролей.

Давалось дерево из N вершин с весами вершин w_i.

Требовалось ответить на Q запросов u_j v_j: какой минимальную сумму весов на пути u_j - v_j можно получить с помощью одного 'поворота' пути a - b, если ни a, ни b не должны лежать на пути u_j - v_j.

Поворот 'пути' – это операция сдвига вправо весов вершин – $w_{path_2} = w_{path_1}, w_{path_3} = w_{path_2}, \dots \ w_{path_1} = w_{path_{last}}$.

Задача G. Восстановление паролей. Решение $O(NQ)$

Рассмотрим путь на котором делается запрос a b, а также вершины, которые лежат на пути $a - b$ и смежные с ним.

Из условия следует, что во время вращения один вес добавится в путь а один из весов из пути уйдет.

Для поиска ответа нужно найти такую пару вершин, одна из которых A лежит на пути (у которой есть смежная не лежащая на пути) и другую B которая смежна с одной из вершин пути $a - b$. При этом разница их весов должна быть максимальна.

Пройдемся циклом по вершинам пути $a - b$ и будем хранить минимальный вес вершины типа A и максимальный вес вершины типа B. Далее рассматривая каждую вершину пути e попытаемся построить 'вращаемый' путь, проходящий через e и либо B либо A будем улучшать ответ.

Сложность решения $O(N)$ на запрос.

Задача H. Циклические нуклеотиды.

Было дано дерево с двумя типами ребер 0 или 1.

Нужно было найти количество циклических путей, который "идут" только вниз.

Циклический путь – путь между двумя вершинами, в котором типы ребер образуют последовательность из повторяющихся сегментов (k единиц после которых идут k нулей):

$$1111000011110000$$

Задача H. Циклические нуклеотиды. Решение 1 $O(N^2)$

Пройдем по дереву DFS, в котором будем подниматься вверх группами по k 0/1.

Заметим, что на каждую вершину затрачивается время $O(N/k)$ – где k – размер группы.

Поэтому в худшем случае получается асимптотика $O(N^2)$.

Задача H. Циклические нуклеотиды. Решение 2 $O(N^2)$

Рассмотрим решение с помощью ДП.

Переберем длину k и для каждой длины посчитаем ДП:

- $f_v = f_p + 1$ – если p является $2k$-м предком v и путь является сегментом.
- $f_v = 0$ – иначе.

Ответом для итерации будет сумма всех f_i, т.е. $\sum_{i=1}^{N} f_i$. Общий ответом является сумма всех ответов для каждой итерации.

В худшем случае требуется на каждой из N итераций пройти по всему дереву за $O(N)$. Поэтому получается асимптотика $O(N^2)$.

Задача H. Циклические нуклеотиды. Решение 3 $O(N\sqrt{N})$

Объединим два алгоритма в одно решение:

- Посчитаем ответ только для $k > \sqrt{N}$ с помощью первого решения (с помощью обхода дерева), однако подниматься будем если только $k > \sqrt{N}$.
- Для $k <= \sqrt{N}$ – применим второе решение (ДП с итерациями), однако итерации будем делать до $\sqrt{N}$.

Итоговое решение – $O(N\sqrt{N})$

Задача I. Циклические нуклеотиды 2.

Было дано дерево с двумя типами ребер 0 или 1.

Нужно было отвечать на два вида запросов:

- 1 u_j v_j – узнать является ли путь из u_j в v_j циклическим.
- 2 u_j v_j k_j – изменить типы ребер так, чтобы путь из u_j в v_j был циклическим.

Циклический путь – путь между двумя вершинами, в котором типы ребер образуют последовательность из повторяющихся сегментов (k единиц после которых идут k нулей):

$$1111000011110000$$

Задача I. Циклические нуклеотиды 2. Решение $O(Qlog^2 N)$

Определим последовательность как 'почти циклическая', если ее можно достроить справа или слева некоторым количеством 0/1 так чтобы она стала циклической:

$$00\ 111000111000\ 1$$

$$(2, 1, 0), (3, 4, 1), (1, 1, 1)$$

Такие последовательности легко описать в виде 'циклических' последовательностей (которые начинаются не обязательно с 1), каждая из которых описывается тремя числами:

- Размер группы.
- Количество групп.
- Цвет первого элемента.

Задача I. Циклические нуклеотиды 2. Решение $O(Qlog^2N)$

Любая “почти циклические” последовательность описывается не более чем тремя 'циклических' последовательностям.

Определим некоторые операции над 'почти циклическими' последовательностями:

- Конкатенация – объединение двух последовательностей.
- Разрез – разделение последовательности на две ко индексу.
- Переворот – реверс всей последовательности.

Операции не всегда могут приводить к “почти циклической” последовательности.

Задача I. Циклические нуклеотиды 2. Решение $O(Qlog^2 N)$

Далее для ответа на запросы можно реализовать Heavy Light декомпозицию с операцией объединения 'почти циклических' последовательностей.

Для обработки запросов:

- 1 u_j v_j – проверить являтся ли путь из u_j в v_j 'циклическим' – делаем запрос на пути из u_j в v_j, и получив "почти циклическую" последовательность проверяем является ли она "циклической".
- 2 u_j v_j k_j – изменить основания нуклеотида из узла u_j в v_j таким образом, чтобы он являлся 'циклическим' – сделаем запрос с групповой модификацией на пути u_j в v_j, при этом для каждого сегмента или спуска по дереву мы будем разрезать исходную 'почти циклические' последовательность.

Обработка каждого запроса $O(log^2 N)$ – время работы Heavy Light декомпозиции.

Значит суммарное время – $O(Qlog^2 N)$.

Задача J. Центрифуга 2.0.

Давалось бинарное дерево из N вершин с корнем в вершине 1 и весами w_i на вершинах.

Требовалось с помощью поворотов ребер сделать вес дерева минимальным, где вес T считается как:

$$T = \sum_{v=1}^{N} w_v * h_v$$

h_v – расстояние от корня до вершины v.

Задача J. Центрифуга 2.0. Решение $O(N^3)$

Выпишем все вершины в порядке обхода DFS – *path*.

Далее посчитаем ДП:

$$f_{l,r} = \min_{l<=c<=r}(f_{l,c-1} + \sum_{i=l}^{r} w_{path_i} + f_{c+1,r})$$

$f_{l,r}$ – лучший способ балансировки вершин с l-ю по r-ю в обходе.

При подсчете будем запоминать вершину $c_{l,r}$ – ту при которой этот минимум выполняется.

Задача J. Центрифуга 2.0. Решение $O(N^3)$

Далее нужно применить вращения таким образом чтобы из начального дерева сделать дерево исходя из массива $c_{l,r}$. Для этого:

- Поворотами ребер исходного дерева сделаем, чтобы ни у одной вершины не было левого потомка.
- Сделаем аналогичные действия с результирующим деревом, но запишем повороты в обратном порядке.

Асимптотика – $O(N^3)$.