МФТИ, ФПМИ

Алгоритмы и структуры данных, осень 2021

Семинар №6. Кучи

1. Определите количество вершин на глубине k в биномиальном дереве порядка n.

2. За сколько работал бы в биномиальной куче `siftDown`?

3. Вместо бинарной (двоичной) кучи можно рассматривать k-ичные кучи. В них каждая вершина (кроме листьев и, возможно, ещё одной вершины) имеет ровно k детей. Покажите, как в такой куче можно реализовать `siftUp` и `siftDown`. Реализуйте через них классические операции кучи. За сколько они работают?

4. Заранее известно, что к куче поступит n^a запросов типа `extractMin` и n^b запросов типа `insert` (в произвольном порядке). Считаем, что $0 < a < b$, где a и b — заданные константы. Докажите, что при каждом n можно подобрать такое k, что на все запросы можно будет ответить с помощью k-ичной кучи за $O(n^b)$.

5. Пусть к изначально пустой биномиальной куче поступает n запросов типа `insert`. Докажите, что она обрабатывает их за суммарное время $O(n)$, хотя некоторые запросы требуют $\Omega(\log n)$ операций.

6. На основе биномиальной кучи разработайте столь же мощную структуру (то есть позволяющую отвечать на те же запросы), которая обрабатывает любой запрос `insert` за $O(1)$. *Указание:* рассмотрите избыточный двоичный счётчик, в каждом разряде которого можно хранить 0, 1 или 2.

7. Пусть имеется алфавит из k символов, а также текст над этим алфавитом, который нужно закодировать с помощью нулей и единиц. Известно, что i-я буква входит в текст n_i раз. Код Хаффмана (код с доказуемо самой маленькой длиной получившейся длины закодированного текста) строится следующим образом. Пусть $n_1 \geqslant n_2 \geqslant \ldots \geqslant n_k$. Сначала построим код Хаффмана для букв с частотами вхождений $n_1, n_2, \ldots, n_{k-2}, n_{k-1} + n_k$. Пусть построен код, состоящий из слов $w_1, \ldots, w_{n-1}$. Тогда искомый код Хаффмана задаётся так: $w_1, w_2, \ldots, w_{n-1}0, w_{n-1}1$. Иными словами, две самые редкие буквы "склеиваются", строится более простой код, а потом к общему слову w_{n-1} приписываются разные биты. Постройте такой код за $O(k \log k)$.

1. Ответ равен C_n^k. Докажите!

2. В худшем случае за $\Omega(\log^2 n)$, поскольку текущее значение нужно сравнивать со всеми детьми, которых может быть вплоть до $\log n$.

3. Процедура `siftDown` будет работать за $\Theta(k \log_k n)$ в худшем случае, а процедура `siftUp` — за $\Theta(\log_k n)$.

4. В качестве k подойдёт n^ε для достаточно маленького ε.

5. Время обработки каждого запроса `insert` пропорционально длине максимального блока младших единиц в двоичной записи текущего количества элементов.

6. Поддерживайте следующий инвариант: между соседними двойками в счётчике обязательно есть хотя бы один ноль, а при каждом `insert` младшая двойка раскрывается в 10. Чтобы хранить младшую двойку, можно хранить список всех позиций двоек.

7. С алгоритмической точки зрения всё просто: в куче нужно хранить пары (число вхождений, буква), затем извлекать из кучи два минимума и вставлять новую букву, соответствующую склейке этих букв.