Шестое задание по алгоритмам

Ситдиков Руслан

19 марта 2018 г.

1 Задача

Очередь можно реализовать на двух стеках $leftStack$ и $rightStack$.Поступим следующим образом: $leftStack$ будем использовать для операции $push$, $rightStack$ для операции pop. При этом, если при попытке извлечения элемента из $rightStack$ он оказался пустым, просто перенесём все элементы из $leftStack$ в него (при этом элементы в $rightStack$ получатся уже в обратном порядке, что нам и нужно для извлечения элементов, а $leftStack$ станет пустым.При выполнении операции $push$ будем использовать три монеты: одну для самой операции, вторую в качестве резерва на операцию pop из первого стека, третью во второй стек на финальный pop. Тогда для операций pop учётную стоимость можно принять равной нулю и использовать для операции монеты, оставшиеся после операции $push$. Таким образом, для каждой операции требуется $O(1)$ монет, а значит стоимость операции $O(1)$.

2 Задача

Будем заполнять массив так, чтобы корнвая вершина дерева была в массиве первой. Заполним компоненты дерева в режиме - слева направо. В случае если у верхушки не имеется дочерних, то добавляем в наш массив 3 значения $NULL$, которое символизирует отсутствие дочерних вершин у родителя. Количество вершин у полного троичного дерева при следующем спуске увеличиваестя в 3 раза. Наш массив начинается с 0 элемента, следовательно, можно узнать номера дочерних вершин нашего родителя, умножая его индекс i на 3. Получим самую первую дочернюю вершину нашего родителя. Прибавляя к индексу 1 и 2 получим все три вершины. Чтобы получить индекс родителя i в массиве нужно проверить делится ли индекс нашей дочерней вершины на 3. Если индекс делится на 3, то делим индекс на 3, находим номер родителя i. Если же мы получаем остаток 1 или 2 при делении на 3, то вычитаем его из индекса дочерней точки и делим на 3.

3 Задача

Пускай у нашей верхушки x в двоичном дереве поиска нет правого ребенка и ее предок $y > x$. Значит, что наша вершина x находится на левой ветке вершины y, следовательно возможны 2 случая.
1)x является дочерней вершиной y (если между x и y стоит какой-то

элемент c, то условие, что после x по возрастанию стоит y , будет нарушено).
2) Имеется вершина $m < x$, на правой ветке которой находится элемент x. Так как $m < x$, то m является предком для x, но y в свою очередь является и предком m, следовательно, y является предком для x.

4 Задача

У b есть правый ребёнок. Тогда c - минимальный элемент из правого поддерева b, а такой элемент определяется спуском по правому поддереву b из корня влево (постоянно выбирая меньший элемент), и когда попадается элемент, у которого нет левого ребёнка (нет элемента, меньшего его), то этот элемент и есть минимальный в правом поддереве b. У b есть левый ребёнок. Тогда a - максимальный элемент из левого поддерева b, который определяется аналогично, как и c, спускаясь от корня вправо, и выбирая элемент, у которого нет правого ребёнка.

6 Задача

Итоговое время ожидания T для n человек будет

$$T = nt_1 + (n-1)t_2 + (n-3)t_3 + ... + t_n.$$

Тогда очевидно, что для минимизации T необходимо, чтобы выполнялось $t_1 \leq t_2 \leq ... \leq t_n$. Или иначе, клиенты должны быть отсортированы по времени по возрастанию.Составим массив пар $M[n]$, где его компонент - пара (i, t_i). Отсортируем его по времени процедурой *Heapsort*. Тогда полученный массив определит последовательность первых элементов пары - последовательность клиентов.Сложность нашего алгоритма $O(nlogn)$.