

Алгоритмы 6

Максим Пасько

17 марта 2018 г.

Задание 1

Пусть в нашем распоряжении есть две очереди - *Left* и *Right*. Нам необходимо реализовать три операции - проверка пустоты (*Empty*), добавление элемента (*Push*), удаление элемента (*Pop*).

```
Empty-stack(Left, Right)
1  if Empty-queue(Left) = true
2      then return true
3  else return false

Push(Left, Right, x)
1  Enqueue(Left, x)

Pop(Left, Right)
1  if size(Left) = 0
2      then error "underflow"
3  else
4      while size(Left) > 1
5          Enqueue(Right, Dequeue(Left))
6      x ← Dequeue(Left)
7      while size(Right) ≠ 0
8          Enqueue(Left, Dequeue(Right))
8  return x
```

Задание 2

Пусть имеется массив элементов $a[n]$. Тогда:

- 1) Корнем дерева выбирается $a[1]$;
- 2) Для элемента $a[i]$ детьми будут элементы $a[3i - 1]$, $a[3i]$, $a[3i + 1]$;
- 3) Для элемента $a[j]$ родителем будет элемент $a[\lfloor \frac{j+1}{3} \rfloor]$.

Преамбула

Алгоритм *Tree-Successor*(x) разделён на две части. Рассмотрим их:

- 1) У x существует правый ребёнок.

В правом поддереве x (назовём его $RST(x)$) содержатся элементы, не

меньшие x . Если x расположен левее какого-либо своего предка y , то все элементы $RST(x)$ будут меньше y , и следующий за x элемент будет минимальный в $RST(x)$. Если же такого y не существует, то только элементы $RST(x)$ будут больше x , соответственно, следующий за x есть минимальный в $RST(x)$.

2) У x нет правого ребёнка.

Левое поддерево x (Назовём его $LST(x)$), если оно существует, содержит элементы, меньшие x , то есть, в $LST(x)$ точно нет следующего за x элемента. Тогда нам необходимо подняться вверх к родителю x . Если мы, поднимаясь вверх от x , найдём какого-либо левого ребёнка какого-то родителя (назовём этого родителя y), то для y справедливы такие высказывания:

$$y > x;$$

Все элементы исходного дерева, за вычетом элементов дерева с корнем y , или больше y , или меньше минимального элемента в дереве с корнем y .

Таким образом, любой элемент дерева или больше y , или меньше $x \Rightarrow y$ - искомый элемент.

Задание 3

Так как у x нет правого ребёнка, то y определяется по пункту 2 преамбулы. То есть, мы шли вверх по дереву от x , пока не нашли левого ребёнка y , что и доказывает требуемое утверждение.

Задание 4

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 (нет элемента, большего его).

Задание 5