

Структуры данных

1. Реализуйте очередь с помощью двух стеков.
2. Продемонстрируйте работу процедуры `Build_Max_Heap` (второе издание Кормена) построения кучи с максимальным свойством на входе

[10, 7, 6, 8, 5, 4, 3, 18, 16].

```

1 Function Build_Max_Heap (a) :
2   | n = a.size;
3   | heap_size(a) = n;
4   | for i =  $\lfloor n/2 \rfloor$  downto 1 do
5   |   | Max_Heapify(a, i);
6   | end
7 end

```

Процедура `Max_Heapify(a, i)` устраняет коллизии, разрешая их начиная с узла, соответствующего *i*-му элементу массива.

3. Сортировка `HeapSort` задана псевдокодом

```

1 Function HeapSort (a) :
2   | n = a.size;
3   | Build_Max_Heap(a);
4   | for i = 1 to n - 1 do
5   |   | s = heap_size(a);
6   |   | a[s] = Extract_Max(a);
7   | end
8 end

```

```

1 Function Extract_Max (a) :
2   | max = a[1];
3   | a[1] = a[heap_size(a)];
4   | heap_size(a) = heap_size(a) - 1;
5   | Max_Heapify(a, 1);
6   | return (max);
7 end

```

Докажите, что сортировка `HeapSort` работает за $\Theta(n \log n)$.

4. На вход задачи подаётся массив пар ключ-значение $[(k_1, v_1), \dots, (k_n, v_n)]$. Необходимо построить бинарное дерево поиска минимальной глубины. Предложите алгоритм, который решает задачу за $\Theta(n \log n)$.

5. Один программист решил хранить двоичное дерево поиска в массиве, используя ту же схему, что и для хранения кучи (во втором издании Кормена). Сколько ячеек массива понадобится ему в худшем случае для хранения дерева с *n* вершинами?

6. Бинарное дерево поиска имеет очень много ключей, поэтому для его обхода в программе можно использовать только стек во внешней памяти, работающий через запросы, и $O(1)$ оперативной памяти (в ней можно хранить конечное число ключей). На вход подаётся последовательность ключей k_1, \dots, k_n , которая помещается в оперативную память. Постройте алгоритм, который возвращает пары (k_i, v_i) для всех ключей последовательности, и при этом совершающий число шагов по дереву, которое отличается от минимально необходимого не больше, чем в два раза.

Примечание. В случае, когда сами ключи находятся недалеко друг от друга их поиск со стартом в корне будет неоптимальным.