1 Задание 8

1.1 Задача 1

Отсортируем с помощью подсчета по первой буквы слова, те слова, у которых совпали первые буквы, отсортируем по второй букве и так далее.

Корректность: алгоритм основывается на сортирвке подсчётом, она работает верно.

Асимптотика: сортировка подсчётом работает за O(n), в худшем случае её придется применить k раз, т.е. $k \cdot O(n) = O(n)$.

1.2 Задача 2

По сути используем алгоритм бинарного поиска. Берем середину, сравниваем элемент с элементом справа и с элементом справа, если участок монотонный, и допустим, последовательность возрастает, то берем левую половину, а если убывает, то правую, либо же мы нашли максимальный элемент. И так повторяем, пока не найдем.

Корректность: на каждом шаге алгоритм отсекает все элементы, которые меньше, чем граница массива, т.е. ни один из этих элементов не будет максимальным. И проверяем является ли этот элемент максимальным.

Асимптотика: У нас 2 сравнения на каждом шаге, и каждый раз мы делим массив пополам, т.е. получили, что алгоритм работает за $O(logn)$.

1.3 Задача 3

Делим кучу на три части, взвешиваем две кучи, если они одинаковые, то монета в третей, если одна из них легче, то монетка в ней. И повторяем процедуру с кучей, которая меньше весит. И ещё нужно проверить за константу, если на каком-то из шагов n не будет делится на три.

1.4 Задача 4

Используем разрешающие дерево.

1. Пусть будем делить пополам, т.е. это будет бин-поиск, который работает за $log_2 n + c$, т.е. медленее

2. Допустим будем делить на большее количество куче, чем три, т.е. нам будет нужно большее количество взвешиваний каждом шаге: если k –

четное, то кол-во взвешиваний: $(\frac{k}{2})log_k n + c = (\frac{k}{2log_3 k})log_3 n + c > log_3 n + c$, если k – нечетное, то кол-во взвешиваний будет равно: $(\frac{k-1}{2})log_k n + c = (\frac{k-1}{2log_3 k})log_3 n + c > log_3 n + c$.

3. Любая комбинация делений будет затрачивать больше взвешиваний, т.к. глубина разрешающего дерева будет минимальна при делении на три кучки. Т.е. $3^h \leq n \to h \leq log_3 n$. С учётом всех округлений получим, кол-во операций: $log_3 n + c$.

1.5 Задача 5

Сравниваем медианы двух массивов m_1 и m_2. Не умаляя общности, будем считать, что $m_1 > m_2$, тогда медиана объединенного массива точно не лежит в правой части первого массива, т.к. m_1 точно больше, чем половина элементов в 1 массиве и половина элементов во 2 массиве и медиана массива, точно не лежит в левой части второго массива, т.к. все элементы из этого подмассива меньше, чем m_1 и m_2, т.е. половина элементов. Тогда сделаем тоже самое для левой части первого массива и правой части массива. Продолжаем так делать, пока не останется по одному элементу в каждом массиве, и выберем наибольший из них.

Корректность: на каждом шаге мы будем отбрасывать элементы массива, которые не могут быть медианой, т.е. если алгоритм выведет некорректный ответ, то он на каком-то шаге выкинул медиану, но противоречит принципу работы описанного алгоритма.

Асимптотика:

1. По времени: на каждом шаге мы уменьшаем каждый из массивов на два, и делаем одно сравнение. Т.е. всего операций будет $1 \cdot log_(n)$. Т.е. асимптотика $\Theta(log_2 n)$.

2. По памяти: кроме хранения массивов нам ничего не требуется, кроме хранения самих массивов. Если только пару переменных для передачи концов и начал подмассивов, т.е. $O(n)$ для хранения массивов.

1.6 Задача 6

а)

в) не in-place
в) Да, она in-place и устойчивая
г) В

1.7 Задача 7

1.8 Задача 8

Если $a_0 > y$, то решений не будет. Иначе $a_0 \le x \le [f(y^{\frac{1}{n}})]$, где $f(x)$ – заданная функция. Далее воспользуемся бин-поиском, смотрим, если $[f(\frac{a_0+[f(y^{\frac{1}{n}})]}{2})] > y$, то берем правый отрезок, иначе берем левый. Округляем левую границу вверх, нижнюю – вниз, т.к. x – натуральное число. Делать мы так можем, потому что функция монотонно возрастающая. Делаем так, пока отрезок не будет длины 1, т.е. между концами не будет целый чисел – только они сами. Если на одном из этих $x, y = f(x)$, то будет существовать, иначе не будет.

Корректность: т.к. функция монотонна, то деля область определения пополам и смотря значение на середине мы узнаем слева или справа от этой точки стоит искать x и выбираем ту часть, в которой он может находиться. Округляя границы, мы можем потерять искомый x, но это будет означать, что он не будет являться натуральным.

1.9 Задача 9

Пусть это будет 14 бросков. Алгоритм будем следующий, разделим этажи на группы по 14, 13, ..., 4, 1, начиная с первого этажа эти группы будут уменьшаться вот так. Кинем шарик с 14-ого этажа, если он разобьется, то проходим все этажи с 1 по 13 вторым шариком (т.е. 14 бросков). Если не разбился, то кидаем $14 + 13 = 27$ этажа, если разбился, то проходим с 15 этажа по 27 вторым, шариком (т.е. $1+13$ бросков). Иначе кидаем с $14+13+12$ этажа и повторяем рассуждения.

Теперь докажем, что не может быть меньше. Пусть это будет 13, значит группа, на которую будем бить будет не больше 13, а следующая не больше 12, ... Подобными рассуждениями получаем, что сумма этих групп будет $(13+1) \cdot \frac{13}{2} = 91 < 100$. Значит 13 групп не подходит, т.к. 100 этажей нельзя разбить на такие группы. Алгоритм и его док-во зиждится на том, что если на каком-то этаже разобьётся шарик, то вторым шариком мы проходим все этажи от того, где мы знаем, что шарик разобьётся, до того этажа, где он разбился.