1. Дано n слов длины k, состоящих из ...

Отсортируем слова по разрядам, используя сортировку подсчетом.

Сложность сортировки подсчетом – $\Theta(n)$. В текущем случае получается, что в процессе сортировки мы пройдемся по k разрядам.

В результате получим сложность алгоритма $\Theta(nk)$ – вроде эффективно....

2. Пусть числовой массив...По сути здесь можно просто применить немного модифицированный алгоритм бинарного поиска, а именно на каждом шаге спрашивать про два соседних элемента.

Действительно, если мы узнаем значения двух соседних элементов, то мы сможем однозначно сказать, справа или слева от них находится максимальный элемент.

То есть этот алгоритм будет иметь примерно в два раза больше ходов, чем бинарный поиск, т.е. по ассимптотике он не уступает и работает за $O(logb)$ ходов.

3. Имеется n монет, среди которых одна фальшивая...

1. Разбиваем все монеты на 3 равные части (если число монет не делится на 3, то количество взвешиваний увеличится на константу, что допустимо в нашем алгоритме, так что с этого момента будем считать, что монеты на 3 кучки у нас делятся всегда вплоть до окончания этого номера.)

2. Далее взвешиваем первые 2 части и забираем себе ту, которая легче. Если первые 2 части оказались равны по массе, то там нет фальшивой монеты, т.е. фальшивка в 3-ей части монет – в этом случае берем ее.

3. Берем все монетки из бывранной кучи и повторяем те же действия.

В результате каждого шага количество рассматриваемых монет уменьшается в 3 раза (+константа). Значит, алгоритм справится с поиском минимальной монеты за $\Theta(log_3 n + c)$.

4. Докажите, что ...необходимо $\log_3 n + c$ взвешиваний.

На каждом шаге после получения результатов взвешивания мы можем сделать соответствующие выводы и понять, что фальшивая монета находится либо среди одной из двух кучек, либо среди монет, не участвоваших во взвешивании. То есть если составить дерево запросов, то на каждом уровне будет ровно три разветвления, откуда следует, что число листьев равно 3^h, где h – высота дерева.

Листья должны перечислить все возможные варианты, т.е. $3^h \geqslant n \Rightarrow h \geqslant log_3 n$.

Минимальное количество запросов как раз ровно высоте дерева. В идеале, когда количество монет – степень тройки, мы имеем, что минимальная высота равна $log_3 n$, т.е. минимальное число запросов равно $log_3 n$.

5. Даны два отсортированных массива длины n ...

1) Поскольку массивы уже отсортированы, то за $O(1)$ находим их медианы и сравниваем между собой. Для определенности будем считать, что мы получили $M_1 > M_2$, где M_1 и M_2 – медианы первого и второго массивов соответственно.

2) Т.к. $M_1 > M_2$, то можем считать, что все элементы второй половины первого массива больше медианы, а все элементы первой половины второго массива меньше медианы, поэтому от этих "кусков"мы можем избавиться. Таким образом, суммарное количество элементов уменьшилось вдвое.

3) Повторяем те же действия (уменьшая на каждом шаге кол-во рассматриваемых элементов в 2 раза), пока не останется 2 массива единичной длины.

4) Сравниваем оставшуюся пару элементов и узнаем положение медианы.

Таким образом, количество сравнений равно $\lceil log_2 n \rceil$. Ассимптотика $O(\log n)$

6. Для каждой сортировки из списка определите, является ли она устойчивой и in-place:
а) QuickSort; MergeSort; InsertionSort; HeapSort.

Устойчивыми сортировками среди перечисленных являются: MergeSort и InsertionSort
in-place сортировки: QuickSort, InsertionSort, HeapSort

Ответы следуют непосредственно из описания алгоритмов, а как сказал Александр Александрович:

Формальное определение алгоритмов можно найти в книге Кормена и др.

7. Определите, что число является значением данного многочлена...

Здесь, вероятно, стоит применить что-то похожее на бинарный поиск, а именно сначала вместо x подставить $[\frac{y}{2}]$. Если значение многочлена при этом больше y, то вместо x подставляем $\frac{y}{4}$, если меньше – $\frac{3y}{4}$, а если равен – то мы победили (число является значением многочлена).

И продолжаем наш "бинарный поиск не забывая, что x – натуральное число. За $\lceil log_2 n \rceil$ шагов мы сможем понять, является ли наш игрек значением многочлена.

Также стоит учесть, что значение многочлена вычисляется не бесплатно! Чтобы вместо каждого икса подставить какое-то число и посчитать значение, требуется $O(n)$ действий. То есть каждый шаг нашего алгоритма имеет сложность $O(n)$.

Таким образом, итоговая сложность у нас получается $O(n \log n)$.

8. Ваш лектор по алгоритмам...

З.Ы. баянистая задача из олимпиадной математики примерно 8-ого класса.

Допустим, нам необходимо n бросков, чтобы определить прочность.

В таком случае мы бросаем 1-ый шарик до n раз. Если на i-том броске он разбивается, то на второй шарик остается $n - i$ бросков.

В итоге получается сумма $100 \leqslant \sum_0^{n-1}(n-i) \Rightarrow n = 14$ – **минимальное число бросков, при котором мы гарантированно узнаем прочность**.

Если же говорить о минимально возможном количестве бросков для нахождения прочность, то получится, что мы сможем найти его, если оба шарика разобьются (к примеру, первый разбился на 14-ом этаже, а второй на первом). То есть число бросков в этом случае будет равно 2.