Филиппова Анастасия

Домашнее задание 5

(1) Создадим массив, в котором каждой букве поставлен в соответствие ее номер в алфавите.
Представим вход (n слов длины k) как матрицу $n \times k$.

$$\begin{pmatrix} a_{11}\ a_{12}\ ...a_{1k} \\ . \\ . \\ . \\ a_{n1}\ a_{n2}\ ...a_{nk} \end{pmatrix}$$

Каждая строка - слово. Символ в строке - буква.
Так как каждо букве мы ставим в соответствие ее номер в алфавите, создадим массив размера $n \times k$, заполним его поставив каждой букве из массива слов цифру (закодируем каждое слово). Затем возпользуемся по разрядной сортировкой $RadixSort$ и получим отсортированный массив цифр. Далее раскодируем слова обратно. Получим отсортированный массив слов.
$RadixSort$ работает линейно от длины входа - $\Theta(nk)$ + кодирование слов и переход от слов к цифрам $2nk$ обращений к массиву букв. Получаем $\Theta(nk)$ - линейно от длины входа. Дополнительно необходимо выделить память под массив размером $n(k+1)$.

(2) (a) Спрашиваем значение $a[1]$ и $a[n/2]$, сравниваем. Если $a[1] > a[n/2] \Rightarrow$ спрашиваем $a[n/4]$ и сравниваем с $a[n/2]$, иначе - $a[n - n/4] = a[3n/4]$ и сравниваем с $a[n/2]$. На каждом следующем шаге размер части массива, на котором происходит поиск $a[t]$ элемента, уменьшается в 2 раза. Следовательно, достаточно $\log_2 n$ сравнений. Предположим, что сделали $K < \log_2 n$ и нашли нужный элемент. $2^K < n$. Каждое число мы "кодируем"теме ответами, которыми получаем (0 - нет, 1 - да). Если $2^K < n$, то по принципу Дирихле получаем, что 2 кода либо совпало, либо про какое-то количество элементов ничего неизвестно. Следовательно, если получать меньше чем $\log 2_n$ ответов - невозможно найти нужный элемент.

(3) На первом шаге делим n монет на 4 группы : в трех $[n/3]$ элементов, в оставшейся $n \ mod\ 3$. Положим на чашечные весы 1-ую и 2-ую группы. Если равны, значит, фальшивая в 3-ьей или 4-ой группах. Если одна из групп легче - значит фильшивая в ней. На следующее взвешивание "проходит"саамая легкая группа и та, в которой было $n \ mod\ 3$ монет. Далее действуем аналогично. В группе оставшихся монет (4-ой) будет всегда $0, 1, 2$ монеты. Таким образом $\lceil \log_3 n \rceil + 1$ - если на последнем шаге группа оставшихся состояла из 1 (в противном случаем $\lceil \log_3 n \rceil$). В итоге $\log_3 n + c$ сравнений достаточно, чтобы найти фальшивую.

(4) Изобразим результаты взвешиваний в виде дерева. Пусть сделали $h < \log_3 n$ взвешиваний. Тогда $3^h < n$. Тогда на последнем уровне в дереве будет 3^h листов, что меньше чем общее число монет, но каждая монета может быть фальшивой. Получается, какие-то монеты не проверили и среди них может быть фальшивая. То есть полученное количество заключений при h взвешиваниях меньше необходимого. Следовательно, необходимо $\log_3 n + c$ взвешиваний (константа возникает из-за округления $\log_3 n$ вверх).

[2]

(5) Псевдо - код:

$l_a, l_b := 0, r_a, r_b := n-1;$
Function $(\mathrm{l}_a, r_a, l_b, r_b)$
if *($l_a == r_a$)* **then**
| return (l_a, r_a, l_b, r_b);
end
if *($a[\frac{l_a - r_a}{2}] > b[\frac{l_b - r_b}{2}]$)* **then**
| $Function(l_a, \frac{l_a - r_a}{2}, \frac{l_b - r_b}{2}, r_b)$;
end
else
| return $Function(\frac{l_a - r_a}{2}, r_a, l_b, \frac{l_b - r_b}{2})$;
end

Ассимптотика:
$T(n) = T(\lceil n/2 \rceil) + O(1)$. При каждом вызове делаем 1 сравнение. Тогда число сравнений $\Theta(\log_2 2n)$
Корректность:
На первом шаге обращаемся к $a[n/2]$ и $b[n/2]$. Если $a[n/2] > b[n/2]$, то $a[n/2]$ точно больше чем n элементов объединения, а $b[n/2]$ точно меньше чем n элементов объединения. Так как массивы отсортированы, то медиана может находиться в $a[0]...a[n/2]$ половине массива a и в $b[n/2]...b[n-1]$ половине массива b. В результате, на $\log_2 n$ шаге размер части массива, на котором мы ищем медиану уменьшиться до 1 элемента. Получаем 2 элемента, которые в объединение являются медианами ($2n$ - четно, поэтому элементы не единственный). Доказали корректность алгоритма.

(6)

(7) Сначала найдем самую тяжелую. Для этого устроим "турнир". Сначала разделим монеты на пары, тогда "в следующий тур"будет проходить монета, которая выиграла в паре. Так сделаем $n-1$ сравнение, чтобы найти самую тяжелую монету. Чтобы найти самую легкую монету достаточно сделать $\lceil n/2 \rceil - 1$ сравнение. Действительно, ведь самая легкая монета не должна выиграть ни разу, следовательно надо выбирать только из монет, проигравших на первом этапе, а их $\lceil n/2 \rceil$