

# Assignment 13, Data-Structures

Oleg Sivokon

*<2016-04-09 Sat>*

## Contents

<b>1</b>	<b>Problems</b>	<b>2</b>
1.1	Problem 1 . . . . .	2
1.2	Problem 2 . . . . .	2
1.3	Problem 3 . . . . .	2
1.3.1	Answer 3 . . . . .	2
1.4	Problem 4 . . . . .	3
1.4.1	Answer 4 . . . . .	3
1.4.2	Answer 5 . . . . .	3
1.5	Problem 5 . . . . .	4
1.5.1	Answer 5 . . . . .	5
1.5.2	Answer 6 . . . . .	5

# 1 Problems

## 1.1 Problem 1

1. How many comparisons does the **partition** algorithm have to make before it completes on an input of size  $n$ , provides the input is sorted in ascending order?
2. What if the input is sorted in descending order?

## 1.2 Problem 2

Array is said to be almost sorted with error of size  $k$  when for any two elements it holds that if there is more than  $k$  elements between them, then they are in correct order.

1. Modify **quicksort** algorithm to produce almost sorted arrays.
2. What is the running time of this new algorithm?

## 1.3 Problem 3

Show the steps taken by the algorithm **randomized-select** on the sequence  $S = (60, 70, 80, 90, 100, 1, 5, 9, 11, 15, 19, 21, 25, 29, 30)$  for  $k = 5$ .

### 1.3.1 Answer 3

This algorithm will use the same **random-partition** procedure used in **quicksort**:

1. *random-partition*( $S, S_1, S_2$ ), randomly selected 19, hence:  $S_1 = (1, 5, 9, 11, 15, 19)$  and  $S_2 = (60, 70, 80, 90, 100, 21, 25, 29, 30)$ .

2. Since  $|S_1| = 6 > k$  the  $k^{th}$  element must be in  $S_1$ , recurse on it.
3. *random-partition*( $S_1, S_3, S_4$ ), randomly select 5, hence:  $S_3 = (1, 5)$  and  $S_4 = (9, 11, 15, 19)$ .
4. Since  $|S_3| = 2 < k$ , decrease  $k$  by 2 and recurse on  $S_4$ :
5.  $k = k - 2$ .
6. *random-partition*( $S_4, S_5, S_6$ ), randomly select 11, hence:  $S_5 = (9, 11)$  and  $S_6 = (15, 19)$ .
7. Since  $|S_5| = 2 = k$  we are done, the  $k^{th}$  element therefore is 11.

#### 1.4 Problem 4

1. Prove that a sequence  $S$  of length  $n$  has at most three elements that repeat more than  $\frac{n}{4}$  times.
2. Write an algorithm finding all elements which repeat more than  $\frac{n}{4}$  times. Running time  $O(n)$ .

##### 1.4.1 Answer 4

Assume for contradiction that there are four elements in  $S$  that repeat  $\frac{n}{4}$  times. That is exist an element in the sequence that repeats  $\frac{n}{4} + 1$  times. Note that  $3\frac{n}{4} + \frac{n}{4} + 1 > n$  contrary to assumed. Hence, by contraction, there are at most three such elements.

##### 1.4.2 Answer 5

Simple solution involves using hash-table: loop over the sequence, using elements as keys in the hash-table. Increment the key whenever you encounter the element. Finally, loop over the hash-table to collect all those for which the condition holds.

This can be improved by using  $\frac{n}{4} \lfloor$  hash-tables, first for the elements which have been seen just once, second—for the elements seen twice and so on. Thus, instead of incrementing the counter, the elements would be promoted to the last hash-table. This removes the requirement of the final loop, but doesn't change the running time significantly.

A more complicated way to do this, without using hash-tables is to do the following:

1. Partition in three parts using the same partition procedure used in **quick-sort** algorithm. If a partition is smaller than  $\frac{n}{4} \lfloor$ , throw it away and repeat.
2. At this point, the following could happen:
  - You have three partitions each containing the same element—you are done.
  - Further partitioning is impossible: no such elements exist—you are done.
  - Some partitions only contain same elements (store the element, throw away the partition), others contain different element: throw them away (they aren't candidates).

## 1.5 Problem 5

1. Given a sequence of real numbers:  $(a_0, a_1, a_2, \dots, a_n)$  define:

$$m = \min\{a_i \mid 0 \leq i \leq n\}$$

$$M = \max\{a_i \mid 0 \leq i \leq n\}$$

Show that there exist  $a_i, a_j$  s.t.:

$$|a_i - a_j| \leq \frac{M - m}{n}$$

2. Write the algorithm for finding the  $a_i, a_j$  from the question above.

### 1.5.1 Answer 5

Note that the sequence with the given  $M$  and  $m$  will have  $M - m = k \geq n$  elements. Suppose now we arrange the elements in increasing order. Suppose, for contradiction, that none of the elements of the sequence satisfies  $|a_i - a_j| \leq \frac{M-m}{n}$ , then it also means that the difference between every two adjacent elements must be at least  $\frac{M-m+\epsilon}{n}$  for some positive  $\epsilon$ . since there are  $k$  pairs of adjoint elements, we get that:

$$\begin{aligned}\sum_{i=1}^k (a_i - a_{i-1}) &= \sum_{i=1}^k \left( \frac{M - m + \epsilon}{n} \right) \\ &= \frac{k}{n} (M - m) + k\epsilon \geq M - m\end{aligned}$$

contrary to assumed. Hence, by contradiction, there must be at least one pair  $a_i, a_j$  s.t.  $|a_i - a_j| \leq \frac{M-m}{n}$ .

### 1.5.2 Answer 6

The general idea for the algorithm is to normalize all members of the given sequence by subtracting the minimum and multiplying by the ratio of one less than the length of the sequence and the difference between the maximum and the minimum. Once done, do the insertion sort: two elements which fall in the same cell will have a distance between them less than the one between the minimum and the maximum divided into one less than the number of elements.

---

**Algorithm 1** Find  $x, y \in Elts$  s.t.  $|x - y| \leq \frac{\max(Elts) - \min(Elts)}{|Elts| - 1}$

---

```

procedure min-pair(elements)
  max  $\leftarrow$  max(elements)
  min  $\leftarrow$  min(elements)
  size  $\leftarrow$  size(elements)
  copy  $\leftarrow$  make-vector(size, nil)
  for element  $\in$  elements do
    index  $\leftarrow$   $\lfloor \frac{(\text{elt} - \text{min}) \times (\text{size} - 1)}{\text{max} - \text{min}} \rfloor$ 
    if copyindex = nil then
      copyindex = element
    elsereturn element, copyindex
    end if
  end for
end procedure

```

---