

CSE2310 Algorithm Design Digital test – part 1

December 10, 2019, 120 minutes

- Usage of the book, notes or a calculator during this test is not allowed.
- This digital test contains 3 questions (worth a total of 10 points) contributing 10/20 to your grade for the digital test. The other digital test contributes the other 10/20 to your grade. Note that the final mark for this course also consists for 40% of the unrounded score for the written exam (if both ≥ 5.0).
- Provide your answers by submitting an implementation solution for the respective assignment on WebLab.
- When an implementation is asked, please provide the **most efficient** implementation in terms of asymptotic time complexity. Providing a suboptimal implementation can lead to a subtraction of points.
- The spec tests in WebLab **do not represent your grade**.
- There is an API specification of the Java Platform (Standard Edition 8) available at <https://weblab.tudelft.nl/docs/java/8>.
- The material for this tests consists of module 1 (Greedy Algorithms) and 2 (Divide and Conquer) of the course and the corresponding book chapters and lectures.
- Total number of pages (without this preamble): 1.
- Exam prepared by M. de Weerd and S. Hugtenburg © 2019 TU Delft.

1. ($2\frac{1}{2}$ points) During the Absolutely Delightful (AD) festival, a number of internationally renowned speakers have been asked to give a presentation about their research. Since the festival attracts thousands of visitors every year, there are several presentations going on at the same time. Thus Phoenix and Mia have to choose what presentations they want to attend. Since they have no idea about the topics, they decide to simply go to as many presentations as possible.

Given the names of n speakers, $name_1, \dots, name_n$, the start times of their presentations s_1, \dots, s_n , and the end time of their presentations e_1, \dots, e_n , return a largest set of names of speakers whose presentations they can attend.

You may assume that they can instantly go from one presentation to another: when a presentation ends at time 8 and the next starts at time 8, this forms no problem. They aren't big on coffee breaks anyway.

2. During the latest Monthly Master Meetings, Mathijs and Stefan have been exchanging notes containing questions to ask the presenter. To ensure the presenter doesn't get a chance to prepare for these questions, nor for the others to steal their questions, they have decided to encode their messages in binary.

Since both Mathijs and Stefan suffer from bad handwriting, it's in everyone's interest if their messages are as short as possible and if they do not need spaces to separate words. To this end they use a Huffman encoding. As you (should) know, 1) this type of binary encoding is a prefix encoding which assigns a binary code to every symbol in the alphabet such that, given the frequency of the symbols occurring in the text, the average bit length is minimized, and 2) a prefix binary encoding can be represented by a binary tree.

For example Figure 1 represents an optimal encoding for the three symbols a, b, c , with frequencies $f(a) = 0.15$, $f(b) = 0.25$, $f(c) = 0.6$. The encodings for these symbols are $f(a) = 00$, $f(b) = 01$, $f(c) = 1$ (this is based on the paths from the root to the nodes, going left appends a 0, going right adds a 1). Note that due to the way that a Huffman encoding merges two symbols with a low frequency, the symbol with the highest frequency (c) has the shortest encoding.

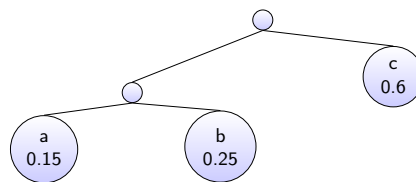


Figure 1: A Huffman (optimal prefix) encoding for three symbols with the given frequencies.

Both of the following implementation tasks are done in the same assignment in WebLab, where they are represented as two different methods to implement. For both tasks you may assume the alphabet has two or more symbols.

- (a) (1 point) Given a node in a tree representing a Huffman encoding of an alphabet, give the encoding of the symbol in that node. E.g. in Figure 1, given the node for 'b' you should return '01'.
 - (b) (4 points) Given n symbols c_1, \dots, c_n with a frequency of f_1, \dots, f_n , respectively, return the (root of a) tree representing an optimal Huffman encoding for this alphabet.
3. ($2\frac{1}{2}$ points) Mathijs and Stefan are analyzing the use of the skill circuits in the course on Algorithm Design. To this end they have been provided with a *sorted* list of skills. The first $s \geq 0$ entries of the list are skills that have not been completed, followed by $t \geq 0$ skills that have been completed. We aim to find this value t . Stefan immediately suggests that they can just iterate over the list and count the number of completed skills in $O(n)$ time where $n = s + t$. But since Mathijs believes Algorithm Design will become very popular, with billions of people following the course, he suggests to use a much more efficient $O(\log n)$ method to find the answer.

Given an integer n and the skills sorted by completion s_1, \dots, s_n , return the number of completed skills. Note that Stefan's "brute-force" $O(n)$ solution is worth zero points.