THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Analysis of Algorithms
(Computer Science 3340b)

ASSIGNMENT 2
Due date: Tuesday, March 2, 2021, 11:55PM ETD

Please **read** general guidelines for answering algorithm design question on **page 3**.

1. In the text book 8.2-1, pp. 196.

2. In the text book 13.3-2, pp. 322.

3. In the text book 13.4-3, pp. 330.

4. Given n elements and an integer k. Design an algorithm to output a sorted sequence of smallest k elements with time complexity $O(n)$ when $k\log(n) \leq n$.

5. Design an **efficient** data structure using (modified) red-black trees that supports the following operations:

$Insert(x)$: insert the key x into the data structure if it is not already there.

$Delete(x)$: delete the key x from the data structure if it is there.

$Find\_Smallest(k)$: find the kth smallest key in the data structure.

What are the time complexities of these operations?

6. In the text book, 21.4-2, pp. 581.

7. In the text book, 21.4-3, pp. 581. And in question 10 a), is it enough to use one byte to store $rank$? Explain your answer.

8. In the text book, 16.3-5, pp. 436.
Hint: show that for an encoding T, if $d_T(c_1) > d_T(c_2)$ and $f(c_1) > f(c_2)$, then T is not optimal.

9. (optional for bonus credit) In the text book, 16.3-6, pp. 436.

10. Next page.

10. (programming question) Finding connected components in a binary image.

a) A Disjoint-Set data structure should be implemented, with the most efficient algorithm (union by rank and path compression), as an abstract data type (a class in C++ or java) with the following operations.

- $uandf(n)$: constructs an disjoint-set data type with n elements, $1, 2, \ldots, n$.
- $make\_set(i)$: creates a new set whose only member (and thus representative) is i.
- $union\_sets(i, j)$: unites the dynamic sets that contains i and j, respectively, into a new set that is the union of these two sets.
- $find\_set(i)$: returns the representative of the set containing i.
- $final\_sets()$: returns the total number of current sets and finalizes the current sets: (i) $make\_set()$ and $union\_sets()$ will have no effect after this operation and (ii) resets the representatives of the sets so that integers from 1 to $final\_sets()$ will be used as representatives.

b) Design and implement (write a program) an algorithm to find the connected components in a binary image using Disjoint-Set data structure in a).

An ASCII file containing a binary image is available (see girl.img and img_readme) as the input of your program. The output of the program should be the following in this specified order:

1. the input binary image,
2. the connected component image where each component is labelled with a unique character,
3. a list sorted by component size, where each line of the list contains the size and the label of a component,
4. same as 2 with the connected components whose sizes are less than three deleted.

In your gaul account, you should create a directory called "asn2" which contains your **asn2_solution.pdf** for question 1 to question 9 and the description of algorithm and the explanation of correctness and complexity of $final\_set()$ in 10 a) and your algorithm in 10 b); your **program** for question 10; the input file with name **infile**; and the **makefile**. The makefile should be written such that the command "make clean" will remove all the "*.o" files and the command "make" will generate an executable file **asn2** that can be run by typing "asn2 < infile". If you are using Java or Python, you may not need the makefile. In that case, you should have a shell script file, **asn2**, so that by typing "asn2 < infile" your java or python program will run.

You should use unix **script** command to capture the screen of the execution of your program. The **resulting file** should also be in directory "asn2".

Your programs have to be able to run on **compute.gaul.csd.uwo.ca** as our TAs will be marking your programs with this machine.

- To answer a question for designing an algorithm, the following three steps are needed
  1. Describe your algorithm in English (**not** with pseudo code);
  2. Show why the algorithm is correct; and
  3. Analyse the complexity of the algorithm.