THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Analysis of Algorithms
(Computer Science 3340b)

ASSIGNMENT 2
Due date: Thursday, March 2, 2023, 11:55PM EST

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

1. In the text book 8.2-1, pp. 210.

2. Show the red-black trees that result after successively inserting the keys in the order 35, 40, 37, 19, 12, 8 into an initially empty red-black tree.

3. Show the red-black trees that result after the successively deletion of the keys in the order 8, 12, 19, 37, 40, 35 from the final red-black tree of question 2.

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?
   (Hint: for $Find\_Smallest(k)$, what if for each node, we add the information of the tree size for the subtree rooted at the node? Is this helpful when searching to decide to go to left subtree or right subtree?)

6. In the text book, 19.4-2, pp. 540.
   (Hint: use induction on the sequence of operations, check the proof of Lemma (1) in the notes, *not* the number of elements, of a disjoint set)

7. In the text book, 19.4-3, pp. 540. Based on your answer, for question 9 a), is it enough to use one byte to store $rank$? Explain your answer.

8. (optional for extra credit) In the text book, 15.3-5, pp. 439.

9. Next page.

9 (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.
- $find\_set(i)$: returns the representative of the set containing 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.
- $final\_sets()$: returns the total number of current sets, $size$, 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 $size$ will be used as representatives.

b) Design and implement 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 face.img.txt and img_readme.txt) 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, but only the connected components whose sizes are greater than 1 will be printed.

5 Same as 2, but only the connected components whose sizes are greater than 11 will be printed.

In your gaul account, you should create a directory called "asn2" which contains your **asn2_solution.pdf** (for your answers of question 1 to question 8, the algorithm and the explanation of correctness and complexity of $final\_set()$ in 9 a), and your algorithm in 9 b)); your **program** for question 9; 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.