

THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Analysis of Algorithms
(Computer Science 3340b)

ASSIGNMENT 2

Due date: Tuesday, February 24, 2015, 11:55 pm

1. In the text book 8.2-4, pp. 197.
2. In the text book 13.3-2, pp. 322.
3. In the text book 13.4-3, pp. 330.
4. In the text book 13-3 a, pp. 333.
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 k th smallest key in the data structure.

What are the time complexities of these operations?

6. 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 $k \log(n) \leq n$. Can you solve the same problem for general k with time complexity $O(k \log(k) + n)$?
(hint: $k \log(n) \leq k \log(k) + n$)

7. In the text book, 16.3-5, pp. 436.
8. In the text book, 21.4-2, pp. 581.
9. In the text book, 21.4-3, pp. 581.
10. Next page.

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

a) An Union-Find data structure should be implemented as an abstract data type (a class in C++) with the following operations.

- `uandf(n)`: constructs an union-find data type with n elements, $1, 2, \dots, n$.
- `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, finalizes the current sets (`make_set()` and `union_sets()` will have no effect after this operation), and resets the representatives of the sets so that integers from 1 to `final_sets()` will be used as representatives.

b) Design and implement (a program) an algorithm to find the connected components in a binary image using Union-Find 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 component whose size equals to one removed.

In your gaul account, you should have a directory called "asn2" which contains your program, the input file, and a 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, you may not need the makefile. In that case, you should have script file, `asn2`, so that by typing "asn2 < infile" your java programs will run.

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