INDIAN STATISTICAL INSTITUTE

Laboratory Test I (Part II)

M. Tech (CS) - I Year, 2014-2015 (Semester - I)

Data and File Structures Laboratory

Date: 27.08.2014 Total Marks : 140 + 30 = 170 (30 marks for good programming habits)

Note: Follow the file naming convention strictly as mentioned.

You are not allowed to connect and browse the internet during the test. No books and e-books are allowed.

Any instance of malpractice would be dealt with sternly.

(Q1) Write a C program that

(a) reads a file having all integer entries and the number of rows and columns of a matrix as the first two entries. (Let the number of rows and columns be r and c, respectively.) Apart from the entries for row and column, all other elements are either 0 or 1. Your program should take the name of the file as a command line argument. The file would be provided to you during the lab test. [8]

(b) creates a 2D array $\mathcal{A}$ of r rows and c columns of integers dynamically using pointer to pointer. [8]

(c) reads $r \times c$ 0-1 entries from the file into $\mathcal{A}$. [4]

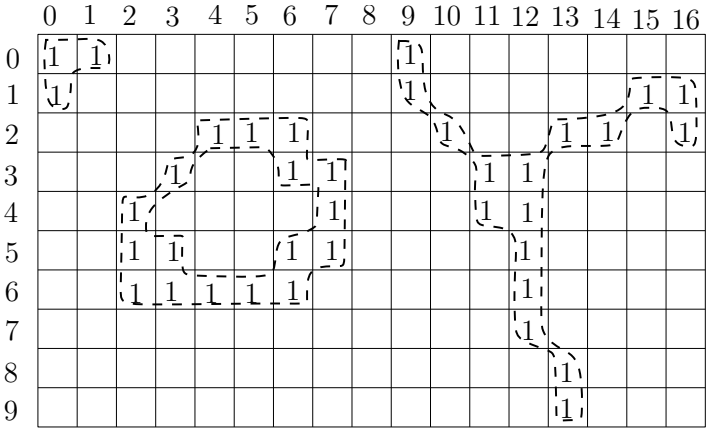

Figure 1: An example with $r = 10$ and $c = 17$. The array entries for only the *pixel*s (i.e., 1s) are shown; the empty cells are all 0s. For the entry $(3, 3)$, you have to report that it belongs to a connected component of size 17 and the entries are $(2, 4)$, $(2, 5)$, $(2, 6)$, $(3, 3)$, $(3, 6)$, $(3, 7)$, $(4, 2)$, $(4, 7)$, $(5, 2)$, $(5, 3)$, $(5, 6)$, $(5, 7)$, $(6, 2)$, $(6, 3)$, $(6, 4)$, $(6, 5)$ and $(6, 6)$. The number of *connected component*s is 3 and the sizes of them are 3, 17 and 17.

(d) has a function that takes as input two indices i and j, and if $\mathcal{A}[i][j]$ is equal to 1, reports the size and the elements in the *connected component* to which $\mathcal{A}[i][j]$ belongs. [25]

We define *connected component* as follows. We term each 1 entry in $\mathcal{A}$ as a *pixel*. The 8-neighborhood of a *pixel* at location (i, j) is the set of locations $(i-1, j-1)$, $(i-1, j)$, $(i-1, j+1)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j-1)$, $(i+1, j)$ and $(i+1, j+1)$ provided they are valid entries of $\mathcal{A}$. A *pixel* p is *connected* to q if q belongs to the 8-neighborhood of p. A *path* from a *pixel* p to a pixel q is a sequence of distinct *pixel*s $p = p_1, p_2, \ldots, p_{n-1}, p_n = q$ such that any two consecutive *pixel*s in the sequence belong to each other's 8-neighborhood. Two *pixel*s are said to be *connected* if there exists a *path* between them. For any *pixel* p in $\mathcal{A}$, the maximal set of *pixel*s that are *connected* to p is called a *connected component* of $\mathcal{A}$. A single *pixel* surrounded by all 0s is a *connected component* of size 1. Note that, by definition, *connected component*s are all disjoint. See Figure 1 for a clarification.

(e) that finds the number of connected components of pixels in $\mathcal{A}$ and the sizes of each connected component. [25]

Total marks: [8+8+4+25+25=70]

(Q2) Write a C program

(a) that reads a positive integer $n \geq 1$ as input and takes as input a set $\mathcal{S}$ of n characters, preferably from command line; [5]

(b) and generates the power set of $\mathcal{S}$. [25]

As an example, for $n = 3$, and the set $\mathcal{S}$ being $\{a, m, n\}$, the elements of the power set of $\mathcal{S}$ are $\{\ \}$, $\{a\}$, $\{m\}$, $\{n\}$, $\{a, m\}$, $\{a, n\}$, $\{m, n\}$, $\{a, m, n\}$.

Total marks: [5+25=30]

(Q3) Suppose, we are given an array of integers with a special property. The array as a whole is not necessarily sorted, but it is divided into blocks of size b (a constant) such that all integers in a block are less than all integers in the next block. For example, the array {3, 2, 5, 8, 10, 31, 23, 11, 32, 45, 40, 39, 49, 48, 47, 60} is one such array with $b = 4$.

(a) In Java or C++, define an interface or abstract class called `BlockArraySearcher`. Also define two classes `BlockSequentialSearcher` and `BlockBinarySearcher` which implement/extend the general interface/class. The interface/superclass should specify a method which takes two inputs: an array of integers (or int) and another integer x. The method should search for x in the array and return the position (index) of x if x is present in the array, -1 otherwise. Do not bother if there are duplicates, it is enough to return one occurrence of x. The interface or abstract class, however, will not provide an actual implementation of this method, but should only define it. [5]

(b) In the two subclasses, implement this method via a sequential search and a binary search respectively. Because the array is not completely sorted, a standard binary search would not work, but you have to adapt the concept of binary search and apply to this particular problem. Do not use more than constant amount of extra space. For example, do not sort the whole array into another array using Java or C++ sort and then apply standard binary search. [5+20]

(c) Now in a test program with a main method, generate a block array (use simple array, `int[]`) of size 10 million with block size 100. Then repeat the following experiment 10 times (in a loop). Generate a random integer in the range of 1 to 10 million, and run the search using both your searcher classes and compare the time taken. In Java, you can use `long start_time = System.currentTimeMillis()` or `System.nanoTime()` to record a timestamp. [10]

Total marks: [5+(5+20)+10=40]