Data Structures and Algorithms

COSC 336 Assignment 7

Instructions.

1. Due date is announced on Blackboard.
2. This is a team assignment. Work in teams as in the previous assignments. Submit one assignment per team, with the names of all students making the team.
3. Your programs must be written in Java.
4. Write your programs neatly - imagine yourself grading your program and see if it is easy to read and understand. At the very beginning present your algorithm in plain English or in pseudo-code (or both). Comment your programs reasonably: there is no need to comment lines like "i++" but do include brief comments describing the main purpose of a specific block of lines.
5. You will submit on Blackboard 2 files. The first file should be a .pdf file with the solution to the exercises and with descriptions in English or in pseudocode of the algorithms for the programming task you are required to do and the results that you are required to report. Make sure you label the results as indicated below. Also include in this pdf file, the JAVA code of your program so that the grader can make comments on it. The second file will contain the Java sources of your two programs.

   For editing the above document with Latex, see the template posted on the course website.

   assignment-template.tex and

   assignment-template.pdf

   To append in the latex file a pdf file, place it in the same folder and then include them in the latex file with

   ```
   \includepdf[pages=-,pagecommand={},width=\textwidth]{file.pdf}
   ```

   To append in the latex file a .jpg file (for a photo), use

   ```
   \includegraphics[width=\linewidth]{file.jpg}
   ```

Exercise 1. Show similarly to Fig 8.3 on page 198 in the textbook, how RadixSort sorts the following arrays:

1. 34, 9134, 20134, 29134, 4, 134
2. 4, 34, 134, 9134, 20134, 29134
3. 29134, 20134, 9134, 134, 34, 4

Exercise 2. Present an $O(n)$ algorithm that sorts n positive integer numbers $a_1, a_2, \ldots, a_n$ which are known to be bounded by $n^2 - 1$ (so $0 \leq a_i \leq n^2 - 1$, for every $i = 1, \ldots, n$. Use the idea of Radix Sort (discussed in class and presented in Section 8.3 in the textbook).

Note that in order to obtain $O(n)$ you have to do Radix Sort by writing the numbers in a suitable base. Recall that the runtime of Radix Sort is $O(d(n+k))$, where d is the number of digits, and k is the base, so that the number of digits in the base is also k. The idea is to represent each number in a base k chosen so that each number in $\{0, 1, \ldots, n^2 - 1\}$ requires only 2 "digits," so $d = 2$. Explain what is the base that you choose and how the digits of each number are calculated, in other words how you convert from base 10 to the base. Note that you cannot use the base 10 representation, because $n^2 - 1$ (which is the largest possible value) requires $\log_{10}(n^2 - 1)$ digits in base 10, which is obviously not constant and therefore you would not obtain an $O(n)$-time algorithm. By the same argument we see that no base k that is constant works, therefore k has to depend on n. In your explanations you need to indicate the formula that gives k as a function of n, and show that $d = 2$ "digits" are enough to represent all the numbers in the range $\{0, 1, \ldots, n^2 - 1\}$.

Illustrate your algorithm by showing on paper similar to Fig. 8.3, page 198 in the textbook (make sure you indicate clearly the columns) how the algorithm sorts the following 2 sequences:

(a) 45, 98, 3, 82, 132, 71, 72, 143, 91, 28, 7, 45.

In this example $n = 12$, because there are 12 positive numbers in the sequence bounded by $143 = 12^2 - 1$.

(b) 45, 98, 3, 82, 132, 71, 72, 143, 91, 28, 7, 45, 151, 175, 145, 399, 21, 267, 346, 292.

In this 2-nd example $n = 20$, because there are 20 positive numbers in the sequence bounded by $399 = 20^2 - 1$.

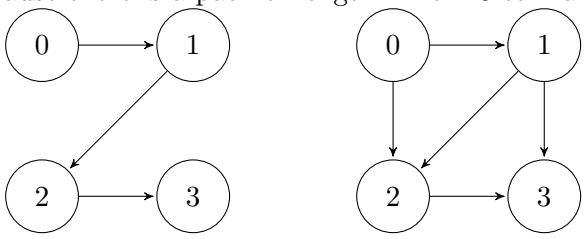
Note: if you use a base b bigger than 10, you do not need to invent symbols for the digits larger than 10; instead use as digits the numbers $0, 1, \ldots, b - 1$ represented in base 10. For instance if you use base, say 25, the digits will be: 0,1, ..., 9, 10, 11, ..., 23, 24. So we view '10', '11', etc., as a single symbol. For instance in this representation the number 9 23 written in base 25 has 2 digits: 9 and 23.

Programming Task.

This is related to Exercise 22.1-5, page 593 in the textbook.

Part (a). Write a program that computes the adjacency list of the directed graph G^2 defined in the exercise (but I also define it just below; this is called the square of G), given the adjacency list of the directed graph G. The promised definition: The graph G^2 has the same nodes as G, and (u, v) is an edge of G^2 if and only if there is path of length 1 or 2 from u to v.

For example if G is the graph on the left, then G^2 is the graph on the right. As you can see G^2 has the edges of G and in our example there are the extra edges $(0, 2)$ and $(1, 3)$, because there is a path of length 2 from 0 to 2 and a path of length 2 from 1 to 3.



For the adjacency list, you **must** use the Java class Adj_List_Graph given in the file Adj_List_Graph.java (see Test_Adj.java for a very simple example of using this class).

You will read the input graph G from a file which contains 2 lines. The first line contains the number n of vertices of the graph. The second line contains a sequence of n^2 bits (values 0 and 1). The n nodes are labeled $0, 1, \ldots, n-1$. If the $i \times n + j$-th bit in the sequence is 1, then there is an edge from node i to node j, and if the $i \times n + j$-th bit in the sequence is 0, then there is no edge from node i to node j. In other words, if the n^2 bits are indexed with indices $0, 1, \ldots, n^2 - 1$, from the bit with index k, you compute $i = k/n$ and $j = k \pmod n$ and if bit with index k is 1, then there exists an edge (i, j), and if it is 0, then there is no edge (i, j). For example, the graph G above has $n = 4$ and the edges are given by the following $n^2 = 16$ bits: 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0.

The program has to create the adjacency list of the graph G^2 and then use the printGraph function of the class Adj_List_Graph to print the edges of G^2.

Run your program on the two data sets from the files

input-7-1.txt

input-7-2.txt

Describe briefly your program and report the results in the .pdf file.