Data Structures and Fundamentals of Programming (Questions 5-8)

Problem 5

Write a function to multiply two sparse matrices represented by a one dimensional array. Assume both matrices are mapped in row major order. The result matrix is similarly represented.

Problem 6

Consider an array $A[1 \cdots n]$ of k consecutive zeros followed by $n-k$ consecutive ones. We want to determine the number of zeros, k.

(a) Write an efficient algorithm to compute k. Analyze its worst-case time complexity.

(b) Write pseudo-code for a recursive function,

`int k(A[ ], left, right)`

that returns the number of zeros, k. Analyze its worst-case time and space complexity.

Problem 7

A multiset is a set of elements in which an element can be repeated multiple time. Here is an example of a multiset.

$$M = \{1, 2, 2, 3, 4, 4, 4, 6, 6, 9, 11, 11, 11, 11, 11, 14, 14, \cdots\}.$$

Consider a data structure for maintaining a multi set M, with the following operations:

- `Init(M)`: creates an empty data structure M.
- `Insert(M, i)`: inserts one copy of i in M.
- `Remove(M, i)`: removes one copy of i in M.
- `Frequency(M, i)`: returns the number of copies of i in M.
- `Select(M, k)`: returns the element in the k-th position of the sorted list M.

In the above example, $Frequency(M, 4) = 3$ and $Select(M, 8) = 6$.

Let $|M|$ be the number of elements in M and $\|M\|$ be the number of different elements in M.

(a) Describe an implementation of the data structure such that $Init(M)$ takes $O(1)$ time and all other operations take $O(\log \|M\|)$ time.

(b) Design a algorithm for sorting a list L in $O(|L| \log \|L\|)$ time using this data structure.

Problem 8

An array A of integers is called *bitonic* if its elements are first strictly increasing and after reaching a maximum value then they are strictly decreasing. More precisely,

$$A[0] < A[1] < \cdots < A[k] > A[k+1] > \cdots > A[n-1] \quad \text{for some } k, 0 \le k \le n$$

Write pseudo-code for a *non-recursive* function that takes a bitonic integer array of size n, and returns the maximum element. The function is required to run in $O(\log(n))$ time, which means you cannot just scan the entire array.