Data structures and algorithms

Consider a two-dimensional $N \times N$ matrix $M[i][j]$, where each row "i" is a vector with $N$ entries ($M[i][j]=1..N$) and there are a total of $N$ rows. Assume that each entry in the matrix stores a positive integer value, and that no two entries in a given row have the same value.

a. [30 \%] Use pseudo-code to describe an algorithm $i=\text{FindLargestRowSum}(M)$ with time complexity $O(N^2)$, which returns $i$ as the index of row $M[i]$ which has the largest sum of all the entries in the row. (If there are multiple rows which have the same largest sum, your function should return the smallest index among them).

b. [40 \%] Use pseudo-code to describe an algorithm $b=\text{SameSet}(i_1,i_2)$ with time complexity $O(N \times \log(N))$ which returns a Boolean $b=\text{TRUE}$ if the set of integers in row $M[i_1]$ is the same as the set of integers in row $M[i_2]$, and returns $b=\text{FALSE}$ otherwise.

c. [30 \%] Use pseudo-code to describe an algorithm $c=\text{SameSetMax}(M)$ that returns the largest number of rows that have the same set of integers (if no rows have the same set of integers, return zero). You can assume that algorithm $b=\text{SameSet}(i_1,i_2)$ as described in part b) has been implemented correctly, and your pseudo-code may invoke it as a procedure call. What is the time complexity of your algorithm?
Consider a single-processor computer with 32-bit virtual and physical address spaces, 2GB physical memory, and fully-associative 16-entry Translation Look-aside Buffer (TLB). The operating system (O/S) supports multi-tasking with processes and virtual memory. The virtual memory subsystem supports paging to disk, and 10GB of hard disk storage is dedicated to virtual memory paging.

Assume the processor can be configured at boot time to support one of two page sizes: either 4Kbytes (small page) or 1Mbyte (large page). Assume the processor uses a two-level page table where: the base address of the first level of the page table is given by a processor Page Table Register Index (PTRI); the 10 most significant bits of the virtual address (VA) are used to index an entry; the data stored in this entry is a memory Pointer to the second level of the page table; the next P bits of the VA are used to index a page table entry (PTE), from which a translated page frame physical address (PA) is obtained:

In all questions below: for full credit, justify your answer, stating your assumptions as needed.

a) [25 %] What is the largest virtual address space size a single process can allocate and use? What is the maximum number of processes that can be timesharing this system, assuming each process allocates and uses the largest possible amount of virtual address space allowed?

b) [25 %] Assume the computer is initialized with 4KByte page size and that there is a single process running on the computer. The process allocates a contiguous array of N Bytes A[0..N-1] and issues a total of KN read and write accesses to byte A[Ri], where each index Ri (i=1..(KN)) is an integer number chosen randomly in the range 0<=Ri<= N-1. Assume K is a very large number, such that K >> N. What is the largest value of N for which the number of page table lookups is a constant for this access pattern?

c) [25 %] Describe two advantages and two disadvantages of configuring the system to use the larger page size versus using the smaller page size.

d) [25 %] Discuss whether and how differences in page sizes may impact the time taken for the O/S to perform a context switch between two processes.