Some CPSC 259 Sample Final Exam Questions (Part 4)
Sample Solutions

DON'T LOOK AT THESE SOLUTIONS UNTIL YOU'VE MADE AN HONEST ATTEMPT AT ANSWERING THE QUESTIONS YOURSELF.

1. {4 marks} What is the worst-case running time for inserting n items into an initially empty hash table, where collisions are resolved by chaining? What if each sequence (chain) is stored in sorted order?

   ANSWER:

   If the sequences are not sorted then the worst-case running time is O(n). If they are stored in sorted order, then the worst-case running time is O(n^2).

2. {4 marks} Suppose that each row of an $n \times n$ array A consists of 1's and 0's such that, in any row of A, all the 1's come before any 0's in that row. Assuming that A is already in memory, describe a function running in O(n lg n) time (*not* O(n^2) time) for counting the number of 1's in A.

   ANSWER:

   To count the number of 1's in A, we can do a slightly modified binary search on each row of A to determine the position of the last 1 in that row (i.e., do a normal binary search until the adjacent neighbor is 0 or it's the end of the list). Then we can simply sum up these values to obtain the total number of 1's in A. It takes O(lg n) time to find the last 1 in each row. Done for each of the n rows, then this takes O(n lg n) time.

3. {4 marks} Let A be a collection of objects. Describe (in words) an efficient function for converting A into a set. That is, remove all duplicates from A. What is the running time of this method? Use Big-O notation.

   ANSWER:

   First, we sort the objects of A using an efficient sorting method with worst-case O(n lg n) time, like Mergesort. Then, we can linearly go through the sorted sequence and remove all duplicates. The duplicates will be side-by-side. It takes O(n lg n) time to sort and O(n) time to remove the duplicates. Overall, this is an O(n lg n) algorithm.

4. {6 marks} Suppose we want to sort an array of size n that contains items that are either 0 or 1.

a) Use Big-O notation to describe the asymptotic worst-case number of comparisons made by Insertion Sort to sort such an array. (No proof is necessary.)
b) Describe a worst-case input for (a), for the number of entries n.
c) Give Big-O notation to describe the asymptotic worst-case scenario for the number of comparisons made by Quicksort to sort such an array. (Again, no proof is necessary.)
d) Describe a worst-case input for (c), for the number of entries n.

ANSWER:

a) Insertion Sort is $O(n^2)$ in the worst case. Since we have no control over the sequence of 0's and 1's, we can't do better than $O(n^2)$ in the general case.

b) Input: 01010101... Note that there are $n/2$ "01" pairs listed here.
or
11...1 00...0 Note that each of these 2 strings contains $n/2$ entries.

c) Quicksort is $O(n^2)$ in the worst case.

d) Input:
000... There are n such entries.
or
111... There are n such entries.

5. {4 marks} In matrix multiplication, the product of two $n \times n$ matrices A and B is the one $n \times n$ matrix C whose i,j-th entry is computed as follows:

$$c_{i,j} = \sum_{k=1 \text{ to } n} a_{i,k} * b_{k,j}$$

If we wrote an algorithm that multiplied matrices in this way, how many scalar multiplications (i.e., " * " operations) would the algorithm perform when multiplying one $n \times n$ matrix by another $n \times n$ matrix? Use Big-O notation. (You do not need to find witnesses; it suffices to provide a Big-O expression, or some expression giving the exact number of scalar multiplications.)

ANSWER:

A single entry $c_{i,j}$ in the result requires n multiplications.

There are $n \times n$ $c_{i,j}$'s in the matrix.

This yields $O(n^2) * O(n) = O(n^3)$ complexity.

6. {5 marks} How do we go about picking a good hash function and hash structure? Provide some general guidelines about what a hash designer needs to think about.

ANSWER:

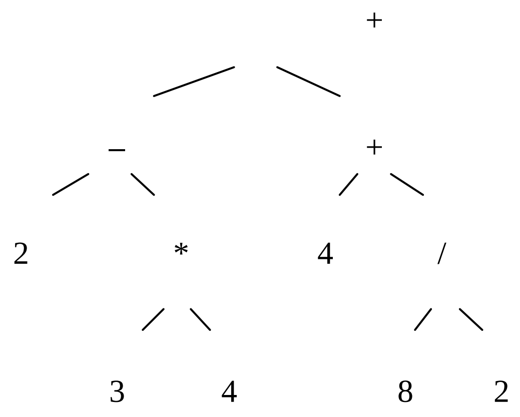
The hash function should distribute the keys uniformly across all possible hash values. Thus, if there are 100 possible hash values (e.g., 100 cells in a hash array using open addressing), then a goal is to distribute approximately 1% of the keys to any one of the given cells. If a sufficiently large number of keys (e.g., 5%+) maps to one of the 100 cells, then a different hash function should be considered.

The hash function should compute its result in O(1) time, which is easy to do, even if dozens of arithmetic steps are involved in the calculation.

Also, the number of cells in the hash structure should be sufficiently large to both hold all the keys and keep the number of collisions low, so that we still get O(1) expected time. When collisions do occur, there should be a good collision resolution policy that tries to minimize clustering, to help achieve O(1) expected time. Don’t make the hash table unnecessarily large, as that wastes memory space.

7. {4 marks} Given the following postfix visitation/traversal of a math expression, draw a valid expression tree for it (similar to the examples from class): 2 3 4 * - 4 8 2 / + +

ANSWER:



8. {2 marks} Explain why quadratic probing is preferable over linear probing, for the open addressing form of hashing.

ANSWER:

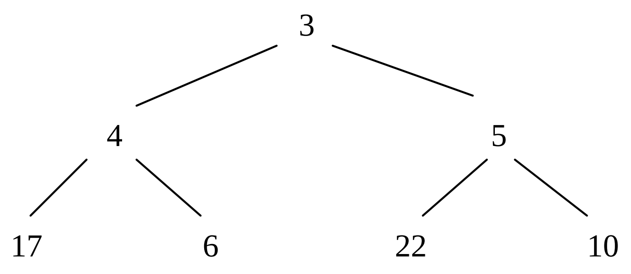
Quadratic probing helps to avoid clustering when doing collision resolution. Thus, two hash probe sequences are less likely to run into each other. The whole point of this is to avoid having the hashing algorithm degenerate into O(*n*) time.

9. {4 marks} Place the following elements into a nearly complete binary tree. After doing that, build a heap using the Heapify algorithm shown in class. The elements (in level-wise order) are as follows:

10, 6, 5, 17, 4, 22, 3

ANSWER:

The final result is:


3
4
5
17
6
22
10