CSCI 2270 Final Exam Review

1. What’s better about breadth first search than depth first search?
   1. Breadth first search is faster than depth first search. Depth first search uses large amounts of memory because it has to store the pointers to each level’s child node while searching that level.
2. Which takes longer, breadth first or depth first search?
   1. Depth first search takes longer because it sorts through all elements.

6. **Hash Tables**: What problem gets worse as more items are added? Collision of different keys into the same slot.

How would your answer change if you used double hashing with a second hash function of modulo 5? Distribute entries to different parts of the table.

How would your answer change if you used chained hashing? Each entry is the head pointer of a linked list, and all collided entries are added to the tail of the list, so multiple entries can fit in the same slot.

7. Explain, in simple English, how a **buffer overrun** hack works.

A program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. C++ provides no built-in protection against accessing or overwriting data in any part of memory and does not automatically check that data written to an array (the built-in buffer type) is within the boundaries of that array. Bounds checking can prevent buffer overflows.

8. What is the difference between a **deep copy** and a **shallow copy**? How can you write a test to tell which one you have? How do pointers and shallow copies relate to each other?

Shallow copy is never a good idea, unless you are copying member variables between things. Shallow copies are bad at storing pointers (head of linked list, first slot of an array). Both items share the same information. If something is changed in object A, object B wont know it’s changed. In a shallow copy, two separate objects are sharing a pointer to the same memory.

1. How can you tell if 2 heaps in array form have all of the same elements?
   * + 1. Check length first. Remove root, reheapify, and look at new root. Keep doing this through every root, and both heaps should be sorted. Then loop through to compare.
2. Why do big\_numbers benefit from a trim() function? When is such a function useful in HW2?
   1. When a number starts with zeros, or when a decimal number ends with zeros. Sum or Diff functions.
3. If we didn’t write big\_number’s operator =, but we used the default version that C++ gives us instead, will we leak memory?
   1. Yes, because it creates a shallow copy. The default = operator is not suited to be used with int, float, etc, not a linked list (big\_num’s data type).
4. Give me an example of the scenario in question 11 causing a crash at runtime.
   1. Using the default = operator with two linked lists of different sizes.
5. Why do we have the rule that heaps must be complete trees?
   1. It is actually an array of values. A tree that is degenerate cannot be an array, so trees cannot be arrays.
6. Given the array 1 4 6 8 3 2 7 5 9 0, show me how quicksort could degrade to quadratic performance in the first 3 partition steps.
   1. Pick worst pivot possible (0, 9) – O(n^2). No advantage of two shorter quicksort partitions afterwards.
7. Given a load factor of 25%, what is the general performance (in terms of expected slots checked) of a doubly-hashed hash table?
8. When can a load factor exceed 100%? Why does this happen?
   1. When a Chained Hash Table is used. It stores multiple keys in a linked list within each key.

Pointer Arithmetic

BST

HEAP

Tree traversal question