**CS311 What we have covered (Yoshii) REVIEW!!!!!!**

* **This list forms the basis of all other classes you will be taking.**
* **Your job is to add a paragraph to each item describing everything you have learned about it.**
* **Put it as the first thing in the CS311 binder.**
* **Then, keep this binder for the rest of your life (or at least until your job interviews are over).**

**“Must Keep” Programs:**

* Stack class
* Queue class
* Linked list classes (with copy constructor and operator overloading)
* Binary search tree class (with Balance Factors)
* Priority queue class
* Graph class
* Hash table class
* Postfix evaluator
* String generator
* DFS of a graph
* MST
* Shortest path

And be able to identify features that contribute to maintainability and reusability in your code.

**Algorithms You Need to be Able to Describe:**

1. Evaluating post fix expressions (stack)
2. Generating all strings made up of a set of characters (queue)
3. Searching through a list
4. Searching through a sorted list
5. Sorting (selection, bubble, insertion)
6. Divide and Conqeuer Sorting (quick, merge)
7. Radix Sort
8. Heap sort (for priority queue)
9. Garbage collection and memory allocation
10. Traversing trees (pre, in, post,)
11. Balancing search trees (rotation, multi-way)
12. Traversing graphs (DFS, BFS)
13. Maximizing the profit in filling a knapsack (Greedy)
14. Finding a minimum tree within a graph (Greedy)
15. Finding the shortest path in a graph (Greedy)
16. Hashing
17. Encryption-Decryption
18. Big Data analysis and Machine Learning

**Complexity Analysis:**

1. Define B(N), A(N) and W(N)
2. Give B(N) and W(N) of the algorithms listed above
3. Give F(N) of sorting and searching of various types
4. Define O, Omega, Theta.
5. Define Static vs. Dynamic
6. Do Time vs. Space decisions in choosing algorithms and data structures
7. Give heights, levels, the number of leaves of binary trees
8. Define a complete graph with N nodes
9. Define a connected graph with N nodes
10. Describe P vs NP and name NP problems

**OOP Features in C++:**

1. Relating classes by composition
2. Relating classes by inheritance
3. Overloading and Copy Constructor
4. Pure interface classes
5. Polymorphism with pointers to objects and virtual functions
6. Friend classes

**CS311 Prep for Final Exam (Yoshii)**

**How to do well on the final:**

* **Be sure to collect all ҉҉ boxes into one file and re-read them.**
* **Be sure to review the Midterm Subst.**
* **Be sure to fill this table before you take the final.**

**Algorithm Type**

**--------------------------------------------------------------**

**Sequential search ---**

How does it work?

worst case # comparisons:

best case # comparisons:

**Binary search ---**

How does it work?

worst case # comparisons:

best case # comparisons:

**What was the lower bound F(n) of search algorithms???**

Unordered list:

Ordered list:

**Selection sort ---**

How does it work?

worst case # comparisons:

[same as best]

best case # comparisons:

[same as worst]

**Bubble sort ---**

How does it work?

worst case # comparisons:

[as many as selection]

best case # comparisons:

[just one pass]

**Insertion sort ---**

How does it work?

worst case # comparisons:

[needed to shift everything to the left for each position 2 through N]

best case # comparisons:

[just needed to look at all positions]

**What was the lower bound for one inversion at a time sorting algorithms?**

**Quick sort -- divide and conquer**

How does it work?

worst case # comparisons:

best case # comparisons:

**Merge Sort -- divide and conquer**

How does it work?

worst case # comparisons:

best case # comparisons:

But the problem is:

**Heap Sort -- tree based/priority queue**

How does it work?

worst case # comparisons:

best case # comparisons:

**Radix sort ---**

How does it work?

# comparisons:

[related to the number of letters you had in each word]

**What was the lower bound of comparison based sorting algorithms??**

**0-1 knapsack Greedy**

How does it work?

worst case # comparisons:

[to sort N ratios ; note: does not produce the best solution]

**MST Greedy**

How does it work?

To Minimize the total cost but keep the graph connected

worst case # comparisons: [total number of fringe vertices]

**Shortest path Greedy**

How does it work?

To Find the shortest path from a vertex to another

worst case # comparisons: [total number of fringe vertices]

**Go over the Final Exam.**

**Course Eval.**

**Thursday: Office Hours only**