CLRS SUMMARY

1. Role of algos

- skip

2. Getting started

- read the whole chapter. it sets notation for the rest of the book.
- don't worry too much about correctness and loop invariants but try to understand. read the complexity analysis for insertion sort. read section 2.3.1 carefully.
- the complexity analysis for merge sort is the template for how almost every other complexity analysis for a recursive function is analyzed.
- must do problems: 2.1-2, 2.1-4, 2.2-2 (and write the algorithm), 2.3-2, 2.3-4 (and write the algorithm), 2.3-5, 2.3-7 (hint: use binary search). do as many of the others that you think you couldn't solve on the spot if someone gave them to you.

3. Growth of functions

- understand the differences between all of the complexity notations, $\Omega, \Theta, O, \omega, o$
- skim the stuff on functions. you really should know it given that you're an adult but it won't kill you (other than that you won't understand some of the complexity analyses later).

4. Divide and conquer

- read about divide and conquer. skim the recurrence methods.
- read the maximum subarray problem closely.
- skim matrix multiplication. look up exponentiation by squaring on wikipedia and read this page https://www.nayuki.io/page/fast-fibonacci-algorithms. this is exactly the kind of thing you'd get asked on an interview.
- skip 4.3, 4.4, 4.5, 4.6
- must do problems: 4.1-1 (if you code it up you'll see), 4.1-2, 4.1-5 (hint: this is called kadane's algorithm. don't look it up until you think about it at least for a day), 4-4 (hint for part a: write out the right hand side multiplying the zs in), 4-5, 4-6

5. Probabilistic analysis

- read 5-1
- skip 5-2
- read 5-3 skim the proofs
- read 5.4.4
- must do problems: 5.1-1, 5.1-2, 5.1-3, 5.3-7 (this is called reservoir sampling [essentially]. look it up and study the priority queue variant), look at 5-1 and 5-2. try to do them.

6. HEAPSORT

- read everything. understand the complexity analyses here - they're basically like merge sort but will come up again for tree algorithms (because a heap is basically a binary tree).
- must do problems: 6.1-1 - 6.1-7, 6.2-2, 6.3-3, 6.5-3, 6.5-6,6.5-7,6.5-8,6.5-9 (hint: the smallest elements of each of the sorted lists should always be in "direct competition"), 6-2a,6-2b, 6-3 (hint for part f: at which corner of the matrix can you be sure of things?)

7. QUICKSORT

- very important. most often used sort in practice.
- look up the implementation of partition on wiki instead though. it's not different, just clearer.
- read the complexity analysis in 7.2. it's important to know why/how quicksort can fail.
- read 7.3
- skim 7.4
- must do problems: 7.1-4, 7.4-3 (because you're an adult), 7-4, 7-5 (important), 7-6

8. SORTING IN LINEAR TIME

- skim 8-1. it's important theory and you should know about the result but no one will ask you to reproduce it
- read 8.2, especially the last paragraph (stability is the reason the last loop goes in reverse - think about why) and 8.3
- figure out how to implement counting sort so that the last loop goes in the forward direction but the sort is still stable
- figure out how to extend counting sort so that it sorts all integers (not just positive numbers).
- implement radix sort
- must do problems: 8.2-3, 8.2-4,8.4-4,8.4-5,8-2e (hint: you should have already figured out how to do this if you implemented radix sort),8-3,8-4ac,8-5a-e

9. MEDIANS AND ORDER STATISTICS

- implement simultaneous max and min
- implement randomized select
- must do problems: 9.1-1 (and code it up), 9.2-3,9.3-5,9.3-6 (and code it up if you really want some exercise),9.3-7,9.3-8 (this one is also a tough implementation problem),9.3-9,9-2