1. CLRS Problem 2-1: Insertion sort on small arrays in merge sort (Note: Not the exercise, but the problem at the end of the chapter)

2. CLRS Problem 2-2: Correctness of bubblesort (Note: Not the exercise, but the problem at the end of the chapter)

3. Finding the Missing Integer - An array A[1 . . . n] contains all the integers from 0 to n except one. It would be easy to determine the missing integer in O(n) time by using an auxiliary array B[0 n] to record which numbers appear in A. In this problem, however, we cannot access an entire integer in A with a single operation. The elements of A are represented in binary, and the only operation we can use to access them is "fetch the jth bit of A[i]," which takes constant time. Show that if we use only this operation, we can still determine the missing integer in O(n) time.

4. Randomized Min - Consider the following algorithm to find the minimum element in an unordered array of n elements:

   1: RandomMin(A[1::n])
   2: m=infinity
   3: for i=1 to n in random order do
      4: if A[i] < m then
      5: m=A[i]
   6: end if
   7: end for
   8: return m

   Notice that the for loop (line 3) takes the numbers 1 to n in random order.
   (a) In the worst case, how many times does RandomMin execute line 5?
   (b) What is the probability that nth (last) iteration executes line 5?
   (c) Analyze the expected number of executions of line 5.

5. VLSI Chip Testing

   You have n supposedly identical VLSI chips that in principle are capable of testing each other. Your test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the answer of a bad chip cannot be trusted. Thus, the four possible outcomes of a test areas follows:

<table>
<thead>
<tr>
<th>Chip A says</th>
<th>Chip B says</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>B is good</td>
<td>A is good</td>
<td>both are good, or both are bad</td>
</tr>
<tr>
<td>B is good</td>
<td>A is bad</td>
<td>at least one is bad</td>
</tr>
<tr>
<td>B is bad</td>
<td>A is good</td>
<td>at least one is bad</td>
</tr>
<tr>
<td>B is bad</td>
<td>A is bad</td>
<td>at least one is bad</td>
</tr>
</tbody>
</table>

   (a) Show that if more than n/2 chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.
   (b) Consider the problem of finding a single good chip from among n chips, assuming that more than n/2 of the chips are good. Show that floor(n/2) pairwise tests are sufficient to reduce the problem to one of nearly half the size.
   (c) Show that the good chips can be identified with Theta(n) (proportional to n ) pairwise tests, assuming that more than n/2 of the chips are good. Give and solve the recurrence that describes the number of tests.
6. CLRS Problem 6-1: Building a heap using insertion (Note: Not the exercise, but the problem at the end of the chapter)

7. A min-max heap is a data structure that permits the retrieval of both the minimum and the maximum elements in a set in O(1) time; the min-max heap can be constructed in O(n) time, while Insert, DeleteMax and DeleteMin take O(log n) time. In a min-max heap, the key of a node at odd level is no larger than the keys of any of its descendants, whereas the key of a node at even level is at least as large as the keys of any of its descendants; consider the root as level 1. Thus, the minimum element in the min-max heap appears at the root, whereas the maximum element appears as one of its two children. Assuming that the min-max heap is stored in an array A in the same manner as an ordinary heap, give an O(log n) implementation of insertion into a min-max heap. (Hint: The basic idea behind insertion is the same as in an ordinary heap.)

8. Sorting Nuts and Bolts - You are given 'n' nuts and 'n' bolts such that each nut ts exactly one bolt. Your only means of comparing these nuts and bolts is with Test(x; y): x is a nut and y is a bolt; returns either "nut is too big," "nut is too small," or "nut fits perfectly." (a) Devise and analyze an O(n^2) worst-case algorithm for matching the nuts and bolts. (b) Devise and analyze an O(n log n) expected-time algorithm for the same problem.

9. Combined-Min - Suppose you have a data structure that supports the following operations:

- make(x): make a new structure D containing only the element x.
- combine-equal(D1;D2): make a new structure D consisting of all the elements of D1 and D2; D1 and D2 must have an equal number of elements.
- delete-min(D): delete the minimum element from D and return its value.

You may assume that make requires O(1) time. (a) Show how to sort any n items with this data structure. (b) Prove that it is impossible for combine-equal to run in time O(n) if delete-min runs in time O(log n). Hint: Recall that comparison-based sorting requires time (n log n).

10. CLRS Problem 5-2: Searching an unsorted array (Note: Not the exercise, but the problem at the end of the chapter)

11. CLRS Problem 7-1: Hoare partition correctness (Note: Not the exercise, but the problem at the end of the chapter)

12. Stooge sort is a recursive sorting algorithm named after the slapstick routines of the Three Stooges, in which each stooge hits the other two. The algorithm is defined as follows: - If the value at the end is smaller than the value at the start, swap them.  
- If there are 3 or more elements in the current list subset, then:  
  Stooge sort the initial 2/3 of the list  
  Stooge sort the final 2/3 of the list  
  Stooge sort the initial 2/3 of the list again

(a) Argue that Stooge-Sort(A, 1, length[A]) correctly sorts the input array A[1..n], where n = length[A].
(b) Give a recurrence for the worst-case running time of Stooge-Sort and a tight asymptotic (T-notation) bound on the worst-case running time.
(c) Compare the worst-case running time of Stooge-Sort with that of insertion sort, merge sort, heapsort, sock sort, and quicksort. Do the professors deserve tenure?

13. CLRS Problem 6-2: Analysis of d-ary heaps (Note: Not the exercise, but the problem at the end of the chapter)
14. CLRS Problem 8-4: Water jugs (Note: Not the exercise, but the problem at the end of the chapter)

15. CLRS Problem 12-1: Binary search trees with equal keys (Note: Not the exercise, but the problem at the end of the chapter)

16. A team of biologists keeps information about DNA structures in a balanced binary search tree (i.e. AVL, red/black, etc) using as key the specific weight (an integer) of the structure. The biologists routinely ask questions of the type: "Are there any structures in the tree with specific weight between a and b (inclusive)?" and they hope to get an answer as soon as possible. Design an efficient algorithm that given integers a and b, returns true if there exists a key x in the tree such that a<=x<=b, and false if no such key exists in the tree. Describe your algorithm in pseudocode or English. What is the time complexity of your algorithm?

17. Give a nonrecursive algorithm that performs an inorder tree walk without using a stack. (Hint: An easy solution uses a stack as an auxiliary data structure. A more complicated, but elegant, solution uses no stack but assumes that we can test two pointers for equality.)

18. CLRS Problem 12-2: Radix trees (Note: Not the exercise, but the problem at the end of the chapter)

19. Double-deletion Heaps - A student suggested the following "extra lazy" variation of the Fibonacci heap called a Double-deletion heap: rather than moving a node up to the root list after it loses its second child, a node is only moved to the root list after it loses its third child. The student discovers that most of the analysis follows as in CLRS, but he cannot bound the maximum degree and comes to you for help. (a) Give a recurrence for the minimum number of nodes in a subtree whose root has degree k. (b) What is a necessary and sufficient condition which would cause Double-deletion heaps to have the same asymptotic amortized running times as standard Fibonacci heaps?

20. CLRS Problem 15-4: Printing neatly (Note: Not the exercise, but the problem at the end of the chapter)

21. Planning a company party - Professor Stewart is consulting for the president of a corporation that is planning a company party. The company has a hierarchical structure; that is, the supervisor relation forms a tree rooted at the president. The personnel office has ranked each employee with a conviviality rating, which is a real number. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to attend. Professor Stewart is given the tree that describes the structure of the corporation. Each node of the tree holds the name of an employee and that employee's conviviality ranking. Describe an algorithm to make up a guest list that maximizes the sum of the conviviality ratings of the guests.

22. Placing Gas Stations Along a Highway - Give an efficient algorithm that on input S, where S={s0=0<=s1<= ... <= si <= ... <= sn=m} is a finite set of positive integers, determines whether it is possible to place gas stations along an m-mile highway such that:
   1. A gas station can only be placed at a distance si in S from the start of the highway. Think of the si as locations of exits on the highway.
   2. There must be a gas station at the beginning of the highway (s0=0) and at the end of the highway (sn=m).
   3. The distance between every two consecutive gas stations on the highway is between 15 and 25 miles (distance equal to 15 or 25 also OK).
   For example, suppose the input is {0, 15, 40, 50, 60}. Then your algorithm should output 'yes', and the distances {0, 15, 40, 60} from the beginning of the highway we can place gas stations. However, if the input is {0, 25, 30, 55, 70}, then your algorithm should output 'no', because there is no subset of the distances that satisfies the conditions listed above. If there is a solution, a valid placement of gas stations, how do you keep track of the locations that the gas stations should go?
23. You are planning purchases of equipment for a business. Bureaucratic problems limit you to one purchase per month, but for each month you delay a purchase, the price of the piece of equipment increases at some rate. You need to purchase n pieces of equipment, E1; E2; : : : ; En where Ei costs $100 if purchased in the rst month, $100*Ri if purchased in the second month, $100*Ri squared if purchased in the third month, and so on (Ri>1 is a rate increase). The pieces of equipment can be purchased in any order. (a) Give an example for which the greedy strategy "lowest rate first" fails to be the cheapest. (b) Prove that the greedy strategy "highest rate first" always gives the cheapest way to schedule purchases. (Hint: If not, when does the strategy make its first mistake?) (c) What is the running time of an algorithm based on the "highest rate first" strategy?

24. Placing Billboards Along a Highway- Suppose you are managing the construction of billboards on a highway of length M miles. The possible sites for billboards are given by increasing mile numbers x1, x2, . . . xn each in the interval [0, M]. If you place a billboard at location xi you receive a revenue of ri>0. Regulations require that no two billboards be within less than or equal to 5 miles of each other. You'd like to place billboards at a subset of the sites so as to maximize your revenue, subject to this restriction. For example, suppose M=20, n=4, {x1, x2, x3, x4}={6, 7, 12, 14} {r1, r2, r3, r4}={5, 6, 5, 1}. Then the optimal solution would be to place billboards at x1 and x3 for a total revenue of 10. Give an efficient algorithm that takes an instance of this problem as input and returns: a) the maximum total revenue that can be obtained from any valid subset of sites, and b) a subset of sites that achieves this maximum total revenue.

25. Single Machine Scheduling - Suppose you have one machine and a set of "n" jobs a1, a2, ..., an to process on that machine. Each job aj has a processing time tj, a profit pj and a deadline dj (all integers). The machine can process only one job at a time, and job aj must run uninterruptedly for tj consecutive time units. If job aj is completed by deadline dj, you receive a profit pj, but if it is completed after the deadline, you receive a profit of 0. It is given that d1<=d2<=...<=dn<=M. Give an efficient algorithm to find the schedule that obtains the maximum amount of profit. That is, for every job selected to be processed, output the starting time of that job.

26. You are given an array of n integers. Consider the problem of finding the maximum sum in any contiguous subvector of the input. For example, in the array: {-6; 12;-7; 0; 14;-7;-3} the maximum sum of 19 is achieved by summing the contiguous elements {12;-7; 0; 14}. Give and explain a THETA(n)-time dynamic programming algorithm for solving this problem. For some partial credit, you may instead give a Theta(n^2)-time algorithm. Explain your algorithm in words rather than just giving pseudocode.

27. Two-Coloring Graphs - Design a linear-time algorithm based on Depth First Seach to 2-color an undirected graph. That is, you must assign the color red or green to each vertex of the graph so that no two adjacent vertices are the same color. If the input graph cannot be so colored, your algorithm must indicate that. Prove your algorithm is correct and that it runs in linear time.

28. BFS and DFS - We are given an undirected, connected graph G = (V;E) and a specific vertex 'u' in V . Suppose that we compute a depth-first forest F rooted at u and a breadth-first tree T rooted at 'u' and obtain the identical structure; that is, F = T. Prove that G = T.

29. CLRS Problem 23-1 parts (a), (b), and (d) Second-best minimum spanning tree (Note: Not the exercise, but the problem at the end of the chapter)

30. CLRS Problem 23-4: Alternative minimum-spanning-tree algorithms (Note: Not the exercise, but the problem at the end of the chapter)
31. Shortest Paths - (a) Suppose we are given a weighted, undirected graph \( G = (V;E) \) and a vertices 's' and 't' in \( V \). Assume that all edges weights are positive. Let \( P \) be a minimum weight path from \( s \) to \( t \). Now we double each edge weight, replacing \( w(e) \) by \( 2w(e) \), thereby creating a new shortest path problem with the same graph but different weights. Is \( P \) necessarily still a minimum-weight path from \( s \) to \( t \)? If true give a proof; if false give a counterexample. (b) Answer part (a) if instead of doubling the edge weights we square them, replacing \( w(e) \) by \( w(e)^2 \).

32. Spanning Tree - Define the "bottleneck" of a spanning tree as its longest edge. Prove that any minimum spanning tree has the smallest possible bottleneck.

33. Optimal Flight Schedules - You want to find the fastest way to \( y \) from Chicago to a set of cities \( C = (c_1; c_2; \ldots; c_n) \) (Chicago is one of the cities, say \( c_1 \)). The program to do this is given the set of cities \( C \) and a set \( F \) of flights, where each flight consists of an origin city, a departure time, a destination city, and an arrival time; all the origin and destination cities are members of \( C \). For each of the cities in \( C \), you want to find a sequence of flights which take you from Chicago to that city in the smallest total travel time. Travel time is measured from the first departure time to the last arrival time. Design and analyze an efficient algorithm that identifies these sequences of flights. To avoid rushing at the airports, your algorithm must leave at least 15 minutes between connecting flights.

34. Bob loves foreign languages and wants to plan his course schedule to take the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are:

- LA15: (none)
- LA16: LA15 is prerequisite
- LA22: (none)
- LA31: LA15 is prerequisite
- LA32: LA16 and LA31 are prerequisites
- LA126: LA22 and LA32 are prerequisites
- LA127: LA16 is prerequisite
- LA141: LA22 and LA16 are prerequisites
- LA169: LA32 is prerequisite

Find a sequence of courses that allows Bob to satisfy all the prerequisites.

35. CLRS Exercise 9.3-9

36. CLRS Problem 19-4: 2-3-4 heaps (Note: Not the exercise, but the problem at the end of the chapter)