1. **How would you compute the parity of a very large number of 64-bit words?:** 5.1 Compute parity
2. **A 64-bit integer can be viewed as an array of 64 bits, with the bit at index 0 corresponding to the least significant bit (LSB), and the bit at index 63 corresponding to the most significant bit (MSB). Implement code that takes as input a 64-bit integer and swaps the bits in that integer at indices i and j.:** 5.2 Swap bits
3. **Write a program that takes a 64-bit word x and returns a 64-bit word consisting of the bits of x in reverse order:** 5.3 Reverse bits
4. **Write a program which takes as input a nonnegative integer x and returns y != x such that y has the same weight as x, and the difference of x and y is as small as possible. You can assume x is not 0, or all 1s.:** 5.4 Find a closest integer with the same weight
5. **Write a program that multiplies two nonnegative integers. The only operators you are allowed to use are assignment and the bitwise operators, i.e. >>, <<, |, &, ~, ^ (You cannot use increment or decrement) You may use loops, conditionals and functions that you write yourself:** 5.5 Compute x*y without multiply or add
6. **Given two positive integers, compute their quotient, using only the addition, subtraction, and shifting operators:** 5.6 Compute x / y
7. **Write a program that takes a double x and an integer y and returns x^y. Assume addition, multiplication, and division take constant time. You cannot use any functions, except for those you write yourself. You can ignore overflow and underflow.:** 5.7 Compute x^y
8. **Write a program that performs base conversion. Assume 2 <= base <= 16. Use "A" to represent 10,..,"F" to represent 15.:** 5.8 Convert base
9. **Implement a function that converts Excel column ids to the corresponding integers, with "A" corresponding to 1. How would you test your code?:** 5.9 Compute the spreadsheet column encoding
10. **Write a program which takes an integer K and returns the integer corresponding to the digits of K written in reverse order.:** 5.10 Reverse digits
11. **Write a program that takes an integer and determines if that integer's representation as a decimal string is a palidrome:** 5.11 Check if a decimal integer is a palindrome
12. **How would you implement a random number generator that generates a random number i between a and b, inclusive, given a random number generator that produces either zero or one with equal probability? All generated values should be equally likely.:** 5.12 Generate uniform random numbers
13. **Write a program which tests if two rectangles have a nonempty intersection. If the intersection is nonempty, return the rectangle formed by their intersection:** 5.13 Check if rectangles intersect
14. **Which doors are open after the 500th person has walked through?:** 5.14 The open doors problem
15. **Design an efficient algorithm for computing the GCD of two numbers without using multiplication, division or the modulus operators:** 5.15 Compute the greatest common divisor
16. **Write a program that takes an array A of length n and an index i into A, and rearranges the elements such that all elements less than A[i] appear first, followed by elements equal to A[i], followed by elements greater than A[i]:** 6.1 The Dutch national flag problem
17. **Write a program which takes as input an array A of digits encoding a decimal number D and updates A to represent the number D+1. For example, if A=<1,2,9>, then you should update A to <1,3,0>:** 6.2 Increment a BigInteger
18. **Write a program that takes two strings representing integers, and returns an integer representing their product.:** 6.3 Multiply two BigIntegers
19. **Write a program which takes an array of n integers, where A[i] denotes the maximum you can advance from index i, and returns whether it is possible to advance to the last index starting from the beginning of the array:** 6.4 Check if a board game is winnable
20. **Implement a function which takes as input as array A of integers and an integer k, and updates A so that all occurrences of k have been removed and the remaining elements have been shifted left to fill the emptied indices. Return the number of remaining elements. There are no requirements as to the values stored beyond the last valid element. For example, if A=<5,3,7,11,2,3,13,5,7> and k=3, then <5,7,11,2,13,5,7,0,0> is an acceptable update to A, and the return value is 7:** 6.5 Delete a key from an array
21. **Write a program which takes as input a sorted array A and updates A so that all duplicates have been removed and the remaining elements have been shifted left to fill the emptied indices. Return the number of valid elements in A.:** 6.6 Delete duplicates from a sorted array
22. **Let A be an array of length n. Design an algorithm to find the smallest positive integer which is not present in A. You do not need to preserve the contents of A. For example, if A=<3,5,4,-1,5,1,-1>, the smallest positive integer not present in A is 2.:** 6.7 Find the first missing positive entry
23. **Design an algorithm that takes a sequence of n three-dimensional coordinates to be traversed, and returns the minimum battery capacity needed to complete the journey. The robot begins with the battery fully charged.:** 6.8 Compute the max difference
24. **What is the maximum profit that can be made by buying and selling a share k times over a given day range?:** 6.9 Generalizations of max difference

25. **Given an array of A of length n whose entries are integers, compute the largest product that can be made using n-1 entries in A. You cannot use an entry more than once. Array entries may be positive, negative, or 0. Your algorithm cannot use the division operator, explicitly or implicitly.:** 6.10 Compute the maximum product of all but one entries
26. **Implement an algorithm that takes as input an array A of n elements, and returns the beginning and ending indices of a longest increasing subarray of A.**
**input: <2,11,3,5,13,7,19,17,23>**
**output: <3,5,13>:** 6.11 Compute the longest contiguous increasing subarray
27. **Write a program that takes a single positive integer argument n (n>=2) and return all the primes between 1 and n:** 6.12 Enumerate all primes to n
28. **Given an array of A of n elements and a permutation P, apply P to A using only constant additional storage. Use A itself to store the result.:** 6.13 Permute the elements of an array
29. **Given a permutation p, return the next permutation under dictionary ordering. If p is the last permutation, return the empty array.**
**input:<1,0,3,2>**
**output:<1,2,0,3>:** 6.14 Compute the next permutation
30. **Design an algorithm for rotating an array A of n elements to the right by i positions. Do not use library functions implementing rotate:** 6.15 Rotate an array
31. **Let a B an array whose entires are all distinct. Implement an algorithm that takes A and an integer k and returns a subset of k elements of A. All subsets should be equally likely. Use as few calls to the random number generator (which returns random integers) as possible and use O(1) additional storage. You can return the result in the same array as input.:** 6.16 Sample offline data
32. **Design an algorithm that creates uniformly random permutations of [0,..,n-1]. You are given an random number generator that returns integers in the set [0,..,n-1] with equal probability; use as few calls to it as possible:** 6.17 Compute a random permutation
33. **Design an algorithm that computes an array of size k consisting of distinct integers in the set [0,..,n-1] All subsets should be equally likely and, in addition, all permutations of elements of the array should be equally likely. Your time complexity should be O(k). Your algorithm can use O(k) space in addition to the k element array for the result. You may assume the existence of a library function which takes as input a nonnegative integer t and returns an integer in the set [0,..,t-1] with uniform probability:** 6.18 Compute a random subset of {0, 1, ..., n-1}
34. **Design an algorithm that reads packets and continuously maintains a uniform random subset of size k of the packets after n >= kth packets are read.:** 6.19 Sample online data
35. **You are given n real numbers and probabilities which sum to 1. Given a random number generator that produces values in [0, 1] uniformly, how would you generate a number in T according to the specified probabilities?:** 6.20 Generate nonuniform random numbers
36. **Check whether a 9x9 2D array representing a partially completed Sudoku is valid. Specifically, check that no row, column, and 3x3 2D subarray contains duplicates. A 0-value in the 2D array indicates that entry is blank; every other entry is in [1,9]:** 6.21 The Sudoku checker problem
37. **Implement a function which takes a 2D array A and returns the spiral ordering of A:** 6.22 Compute the spiral ordering of a 2D array
38. **Implement an algorithm to rotate A, an nxn 2D array, by 90 degrees clockwise. Assume that n=2^k for some positive integer k:** 6.23 Rotate a 2D array
39. **Write a program which takes as input a nonnegative integer n and returns the first n rows of Pascal's triangle:** 6.24 Compute rows in Pascal's Triangle
40. **Write a program which takes as input a 2D array of 1s and 0s, where the 0s encode the positions of rooks on an nxm chessboard and updates the array to contain 0s at all locations which can be attacked by rooks.:** 6.25 Identify positions attacked by rooks
41. **Let F be an nxn Boolean 2D array representing the "knows" relation for n people. F[a][b] is true if and only if a knows b. Design an algorithm to find the ceebrity.:** 6.26 Identify the celebrity
42. **Implement string/integer inter-conversion functions:** 7.1 Interconvert strings and integers
43. **Write a program which takes as input a string s, and removes each "b" and replaces each "a" by "dd". Assume s is stored in an array that has enough space for the final result:** 7.2 Replace and remove
44. **Implement a function which takes as input a string s and returns true if s is a palindromic string:** 7.3 Test palindromicity
45. **Implement a function for reversing the words in a string s:** 7.4 Reverse all the words in a sentence
46. **Write a program which takes as input a phone number, specified as a string of digits, and returns all possible character sequences that correspond to the phone number. The cell phone keypad is specified by a mapping that takes a digit and returns the corresponding set of characters. The character sequences do not have to be legal words or phrases:** 7.5 Compute all mnemonics for a phone number
47. **Write a program that takes as input an integer n and returns n-th integer in the look-and-say sequence.:** 7.6 The look-and-say problem
48. **Write a program which takes as input a valid Roman number string s and returns the integer it corresponds to:** 7.7 Convert from Roman to decimal

49. **Write a program that determines where to add periods to a decimal string so that the resulting string is a valid IP address. There may be more than one valid IP address corresponding to a string, in which case you should print all possibilities:** 7.8 Compute all valid IP addresses
50. **Write a program which takes as input a string s and returns the snake-string of s**
**input: "Hello_World!"**
**output: "e_lHloWrdlo!":** 7.9 Write a string sinusoidally
51. **Implement run-length encoding and decoding functions. Assume the string to be encoded consists of letters of the alphabet, with no digits, and the string to be decoded is a valid encoding:** 7.10 Implement run-length encoding
52. **Let A be an array of n integers. Write an encode function that returns a string representing the concatenation of the Elias gamma codes for A[0],...,A[n-1] in that order, and a decode function that takes a string s assumed to be generated by the encode function, and returns the array that was passed to the encode function:** 7.11 Implement Elias gamma encoding
53. **Implement the UNIX tail command:** 7.12 Implement the UNIX tail command
54. **Write a program which takes as input an array A of strings and a positive integer L, and computes the justification of the text specified by A:** 7.13 Justify text
55. **Given two strings s (the "search string") and t (the "text"), find the first occurrence of s in t.:** 7.14 Find the first occurrence of a substring
56. **Write a program that takes L and F, and returns the merge of L and F. Your code should reuse the nodes from the lists provided as input. Your program should use O(1) additional storage. The only field you can change in a node is next:** 8.1 Merge two sorted lists
57. **Give a linear time nonrecursive function that reverses a singly linked list. The function should use no more than constant storage beyond that needed for the list itself.:** 8.2 Reverse a singly linked list
58. **Write a program which takes a singly linked list L and two integers s and f as arguments, and reverses the order of the nodes from the s-th node to f-th node, inclusive. The numbers begins at 1, i.e., the head node is the first node. Perform the reversal in a single pass. Do not allocate additional nodes.:** 8.3 Reverse a single sublist
59. **Write a program which takes as input a singly linked list L and a nonnegative integer k, and reverses the list k nodes at a time. If the number of nodes n in the list is not a multiple of k, leave the last n mod k nodes unchanged. Do not change the data stored within a node.:** 8.4 Reverse sublists k at a time
60. **Given a reference to the head of a singly linked list, how would you determine whether the list ends in a null or reaches a cycle of nodes? Write a program that returns null if there does not exist a cycle, and the reference to the start of the cycle if a cycle is present. (You do not know the length of the list in advance.):** 8.5 Test for cyclicity
61. **Let L1 and L2 be cycle-free singly linked lists. How would you determine if there exists a node that is common to both L1 and L2.:** 8.6 Test for overlapping lists - lists are cycle-free
62. **Let L1 and L2 be singly linked lists that might have a cycle. How would you determine if there exists a node that is common to both L1 and L2. If such a node exists, return a node that appears first when traversing the lists. This node may not be unique - if L1 ends in a cycle, the first node encountered when traversing L1 may be different from the first node encountered when traversing L2, even though the cycle is the same.:** 8.7 Test for overlapping lists - lists may have cycles
63. **Write a program which deletes a node in a singly linked list. The input node is guaranteed not to be the tail node:** 8.8 Delete a node from a singly linked list
64. **Given a singly linked list and an integer k, write a program to remove the k-th last element from the list. Your algorithm cannot use more than a few words of storage, regardless of the length of the list. In particular, you cannot assume that it is possible to record the length of the list.:** 8.9 Remove the kth last element from a list
65. **Write a program that takes as input singly linked list L of n integers in sorted order, and removes duplicates from it. The list should be sorted. You may perform the change on L itself.:** 8.10 Remove duplicates from a sorted list
66. **Write a program that takes as input a singly linked list and a nonnegative integer k, and returns the list cyclically shifted to the right by k.:** 8.11 Implement cyclic right shift for singly linked lists
67. **Write a program that computes the even-odd merge:** 8.12 Implement even-odd merge
68. **Let L be a singly linked list. Assume its nodes are numbered starting at 0. Define the zip of L to be the list consisting of the interleaving of the nodes numbered 0,1,2,... with the nodes numbered n-1,n-2,n..., where the number of nodes in the list. Implement the zip function:** 8.13 Implement list zipping
69. **Implement a function which takes a postings list and returns a copy of it. You can modify the original list, but must restore it to its initial state before returning.:** 8.14 Copy a postings list
70. **Write a program that tests whether a singly linked list is palindromic:** 8.15 Test whether a singly linked list is palindromic
71. **Write a program that takes a pointer to an arbitrary node in a sorted circular linked list, and returns the median of the linked list.:** 8.16 Compute the median of a sorted circular linked list
72. **Implement a function which takes as input a singly linked list and an integer k and performs a pivot of the list with respect to k. The relative ordering of nodes that appear before k, and after k, must remain unchanged; the same must hold for nodes holding keys equal to k:** 8.17 Implement list pivoting

73. **Write a program that takes as input a singly linked list whose nodes hold integer keys and sorts list:** 8.18 Sort a list
74. **Write a program which takes two singly linked lists of digits, and returns the list corresponding to the sum of the integers they represent. The least significant digit comes first.:** 8.19 Add list-based integers
75. **Design a stack that supports a max operation, which returns the maximum value stored in the stack:** 9.1 Implement a stack with max API
76. **Write a program that takes an arithmetical expression in RPM and returns the number that the expression evaluates to.:** 9.2 Evaluate RPN expressions
77. **Write a program that tests if a string is made up of the characters (,),[,],{,} is well formed:** 9.3 Test a string over "[,],(,),[,]" for well-formedness
78. **Write a program that takes as input a string made up of the characters (,) and returns the size of a maximum length substring in which the parans are matched.:** 9.4 Compute the longest substring with matching parans
79. **Write a program which takes a pathname, and returns the shortest equivalent pathname. Assume individual directories and files have names that use only alphanumeric characters. Subdirectory names may be combined using forward slashes (/), current directory (.) and parent directory (..):** 9.5 Normalize pathnames
80. **Given a BST node, compute al the keys at that node and its descendants. The nodes should be returned in sorted order, and you cannot use recursion.:** 9.6 BST keys in sort order
81. **Write recursive and iterative routines that take a postings list, and computes the jump-first order. Assume each node has an order field, which is an integer that is initialized to -1 for each node.:** 9.7 Search a postings list
82. **Design an algorithm that processes buildings in east-to-west order and returns the set of buildings which view the sunset. Each building is specified by its height.:** 9.8 Compute buildings with a sunset view
83. **Design an algorithm to sort a stack in descending order, i.e., the top of the stack holds the largest value. The only operations allowed are push, pop, top and empty. You cannot explicitly allocate memory outside of a few words.:** 9.9 Sort a stack
84. **Given the root of a binary tree, return an array consisting of the keys at the root and its descendants. Keys should appear in the order of the corresponding nodes' depths.:** 9.10 Compute binary tree nodes in order of increasing depth
85. **Implement a queue API using an array for storing elements. Your API should include a constructor, which takes as argument the capacity of the queue, enqueue and dequeue functions, a size function, which returns the number of elements stored, and implement dynamic resizing:** 9.11 Implement a circular queue
86. **How would you implement a queue given a library implementing stacks?:** 9.12 Implement a queue using stacks
87. **Implement a queue with enqueue, dequeue, and max operations:** 9.13 Implement a queue with max API
88. **You are given traffic at various times and a window length. Compute for each time, the maximum traffic over the window length time interval which ends at that time. The input is specified by an array, where each entry is a timestamp and corresponding amount of traffic:** 9.14 Compute the maximum of a sliding window
89. **Given a positive integer n, how would you compute a shortest straight-line program to evaluate x^n?:** 9.15 The shortest straight-line program for x^n
90. **Write a program that takes as input the root of a binary tree and checks whether the tree is balanced.:** PL 10.1 Test if a binary tree is balanced
    1. Brute force T: O(n) S: O(n)
    2. Early termination T: O(n) S: O(h)
    - return <balanced:bool, height:int>
91. **Write a program that returns the size of the largest subtree that is complete:** PL 10.1.1 Test if a binary tree is balanced
92. **Define a node in a binary tree to be k-balanced if the difference in the number of nodes in its left and right subtrees is no more than k.**

    **Design an algorithm that takes as input a binary tree and positive integer k, and returns a node in the binary tree such that the node is not k-balanced, but all of its descendants are k-balanced. If no such node exists, return null.:** PL 10.2 Find k-unbalanced nodes
    1. Brute force O(n^2)
    2. Brute force + cache T: O(n) S: O(n)
    3. Early termination T: O(n) S: O(h)
    - return <node, count:int>
93. **Write a program that checks whether a binary tree is symmetric:** PL 10.3 Test if a binary tree is symmetric

    1. Brute force T: O(n) S: O(n)
    2. Naïve: T: O(n) S: O(h)
94. **Design an algorithm for computing the LCA of two nodes in a binary tree in which nodes do not have a parent field.:** PL 10.4 Compute the lowest common ancestor in a binary tree

    1. Brute force O(n^2)
    2. Early termination T: O(n) S: O(h)
    - return <state:int, node>
95. **Given two nodes in a binary tree, design an algorithm that computes their LCA. Assume that each node has a parent pointer.:** PL 10.5 Compute the LCA when nodes have parent pointers
    1. Brute force T: O(h) S: O(h)
    2. Equal depth T: O(h) S: O(1)
96. **Design an algorithm to compute the sum of the binary numbers represented by the root-to-leaf paths:** PL 10.6 Sum the root-to-leaf paths in a binary tree
97. **Write a program which takes as input an integer and a binary tree with integer node weights, and checks if there exists a leaf whose path weight equals the given integer:** PL 10.7 Find a root to leaf path with specified sum

98. **Write a program that efficiently compute the k-th node appearing in an in-order traversal. Assume that each node stores the number of nodes in the subtree rooted at that node.:** PL 10.8 Compute the k-th node in an inorder traversal
99. **Write a non-recursive program for computing the in-order traversal sequence for a binary tree. Assume nodes have parent fields.:** PL 10.9 Implement an in-order traversal with O(1) space
100. **Write a program which takes as input a binary tree and performs a per-order traversal of the tree. Do not use recursion. Nodes do not contain parent references. Do the same for a post-order traversal.:** PL 10.10 Implement pre-order and post-order traversals without recursion
101. **Design an algorithm that computes the successor of a node in a binary tree. Assume that each node stores its parent.:** PL 10.11 Compute the successor
102. **Given an in-order traversal sequence and a pre-order traversal sequence of a binary tree write a program to reconstruct the tree. Assume each node has a unique key.:** PL 10.12 Reconstruct a binary tree from traversal data
103. **Design an algorithm for reconstructing a binary tree from a preorder traversal visit sequence that uses null to mark empty children.:** PL 10.13 Reconstruct a binary tree from a preorder traversal with markers
104. **Given a binary tree, compute a linked list from the leaves of the binary tree. The leaves should appear in the left-to-right order.:** PL 10.14 Form a linked list from the leaves of a binary tree
105. **Write a program that computes the exterior of a binary tree:** PL 10.15 Compute the exterior of a binary tree
106. **Write a program that takes a perfect binary tree, and sets each node's next field to the node on its right, if one exists.:** PL 10.16 Compute the right sibling tree
107. **Write the following methods for a binary tree node class:**
**1. A function to test if the node is locked**
**2. A function to lock the node. If the node cannot be locked, return false, otherwise lock it and return true.**
**3. A function to unlock the node.**
**Assume that each node has a parent field. The API will be used in a single threaded program, so there is no need for concurrency constructs such as mutexes or synchronization.:** PL 10.17 Implement locking in a binary tree
108. **Design an algorithm that takes a set of files containing stock trades sorted by increasing trade times, and writes a single file containing the trades appearing in the individual files sorted in the same order. The algorithm should use very little RAM, ideally of the order of a few kilobytes.:** PL 11.1 Merge sorted files
109. **Design an efficient algorithm for sorting a k-increasing-decreasing array.:** PL 11.2 Sort an increasing-decreasing array
110. **Write a program which takes an input a very long sequence of numbers. Each number is at most k away from its current sorted position. For example, no number in the sequence**
**<3,-1,2,6,4,5,8>**
**is more than 2 away from its final sorted position. Design an algorithm that outputs the numbers in the correct order.:** PL 11.3 Sort an almost-sorted array
111. **How would you compute the k stars which are closest to Earth?:** PL 11.4 Compute the k-closest stars
112. **Design an algorithm for computing the running median of a sequence:** PL 11.5 Compute the median of online data
113. **Given a max-heap, represented as an array A, design an algorithm that computes the k largest elements stored in the max-heap. You cannot modify the heap.:** PL 11.6 Compute the k largest elements in a max-heap
114. **Write a program for computing the minimum number of tickets to distribute to the developers, while ensuring that if a developer has written more lines of code than a neighbor, then he receives more tickets than his neighbor.:** PL 11.7 Compute fair bonuses
115. **Design an algorithm to compute the k elements closet to the median of an array:** PL 11.8 Find k elements closest to the median
116. **Design an algorithm for determining whether the k-th largest element in a max-heap is smaller than, equal to, or larger than a given value.:** PL 11.9 The k-th largest element in a max-heap
117. **How would you implement a stack API using a heap?:** PL 11.10 Implement a stack API using a heap
118. **Write a method that takes a sorted array and a key and returns the index of the first occurrence of that key in the array.:** PL 12.1 Search a sorted array for first occurrence of k
119. **Design an efficient algorithm that takes a sorted array and a key, and finds the index of the first occurrence an element greater than that key.:** PL 12.2 Search a sorted array for the first element greater than k
120. **Design an efficient algorithm that takes a sorted array of distinct integers, and returns an index i such that the element at index i equals i.:** PL 12.3 Search a sorted array for entry equal to its index
121. **Design an O(logn) algorithm for finding the position of the smallest element in a cyclically sorted array. Assume all elements are distinct.:** PL 12.4 Search a cyclically sorted array
122. **Design an algorithm that takes a sorted array whose length is not known, and a key, and returns an index of an array element which is equal to the key. Assume that an out-of-bounds access throws an exception.:** PL 12.5 Search a sorted array of unknown length
n,1: brute force, linear search
logn+logn,1: 2^n -> search end then binary search
123. **Write a program which takes a non-negative integer and returns the largest integer whose square is less than or equal to the given integer.:** PL 12.6 Compute the integer square root

124. **Implement a function which takes as input a floating point value and returns its square root.:** PL 12.7 Compute the real square root

125. **You are given two sorted arrays and a positive integer k. Design an algorithm for computing the kth smallest element in array consisting of the elements of the initial two arrays arranged in sorted order. Array elements may be duplicated within and across the input arrays.:** PL 12.8 Search in two sorted arrays

126. **Design an algorithm that takes a 2D sorted array and a number and checks whether that number appears in the array.:** PL 12.9 Search in a 2D sorted array

127. **Design an algorithm to find the min and max elements in an array.:** PL 12.10 Find the min and max simutaneously

128. **Design an algorithm for computing the k-th largest element in an array. Assume entries are distinct.:** PL 12.11 Find the k-th largest element

129. **Devise an algorithm that computes where to place the mailbox so as to minimize the total distance, that residents travel to get to the mailbox:** PL 12.12 Compute the optimum mailbox placement

130. **Design an algorithm for computing the k-th largest element in a sequence of elements:** PL 12.13 Find the k-th largest element - large n, small k

131. **Suppose you were given a file containing roughly one billion IP addresses, each of which is a 32-bit unsigned integer. How would you programmatically find an IP address that is not in the file? Assume you have unlimited drive space but only a few megabytes of RAM at your disposal.:** PL 12.14 Find the missing IP address

132. **You are given an array of n integers, each between 0 and n-1, inclusive. Exactly one element appears twice, implying that exactly one number between 0 and n - 1 is missing from the array. How would you compute the duplicate and missing numbers?:** PL 12.15 Find the duplicate and missing elements
n,n: hashtable
logn,1: sorting
n,1: XOR all = m xor t

133. **Given an integer array, in which each entry but one appears in triplicate, with the remaining element appearing once, find the element appearing once.**
**Input: <2,4,2,5,2,5,5>**
**Output: 4:** PL 12.16 Find an element that appears only once

134. **Write a program that takes as input a set of words and returns groups of anagrams for those words**

**Input: debitcard, elvis, silent, badcredit, lives, freedom, listen, levis**
**Output: <debitcard, badcredit>, <evlis, lives, levis>, <silent, listen>:** PL 13.1 Partition into anagrams

135. **Write a program to test whether the letters forming a string can be permuted to form a palindrome.**
**i.e. edified -> deified:** PL 13.2 Test for palindromic permutations

136. **You are required to write a method which takes text for an anonymous letter and text for a magazine. Your method is to determine if it is possible to write the anonymous letter using the text from the magazine.:** PL 13.3 Is an anonymous letter constructible?

137. **Implement a cache for looking up prices of books identified by their ISBN. You should support lookup, insert, and erase methods. Use the LRU strategy for cache eviction policy.:** PL 13.4 Implement an ISBN cache

138. **Design an algorithm for computing the LCA of two nodes in a binary tree. The algorithm's time complexity should depend only on the distance from the nodes to the LCA.:** PL 13.5 Compute the LCA, optimizing for close ancestors

139. **You are given an array of strings. Compute the k strings that appear most frequently in the array.:** PL 13.6 Compute the k most frequent queries

140. **You are given a set of points in the plane. Each point has integer coordinates. Design an algorithm for computing a line that contains the maximum number of points in the set.:** PL 13.7 Find the line through the most points

141. **Write a program which takes as input an array and finds the distance between a closest pair of equal entries.:** PL 13.8 Find the nearest repeated entries in an array

142. **Write a program which takes an array of strings and a set of strings, and return the indices of the starting and ending index of a shortest subarray of the given array that "covers" the set, i.e., contains all strings in the set.:** PL 13.9 Find the smallest subarray covering all values

143. **Write a program that takes two arrays of strings, and return the indices of the starting and ending index of a shortest subarray of the first array (the "paragraph" array) that "sequentially covers", i.e., contains all the strings in the second array (the "keywords" array), in the order in which they appear in the keywords array. You can assume all keywords are distinct.:** PL 13.10 Find smallest subarray that sequentially covering all values

144. **Write a program that takes an array and returns the length of a longest subarray with the property that all its elements are distinct.**
**input: <f,s,f,e,t,w,e,n,w,e>**
**output: <s,f,e,t,w>:** PL 13.11 Find the longest subarray with distinct entries

145. **Write a program which takes as input a set of integers represented by an array, and returns the size of a largest subset of integers in the array having the property that if two integers are in the subset, then so are all integers between them.**
**Input: <3,-2,7,9,8,1,2,0,-1,5,8>**
**Output: 6 (<-2,-1,0,1,2,3>):** PL 13.12 Find the length of a longest contained range

146. **Write a program which takes as input a string (the "sentence") and a list of strings (a set of "words"), and returns all substrings of the sentence string which are the concatenation of all the words. Each word must appear exactly once, and the ordering is immaterial. Assume all words have equal length.:** PL 13.13 Compute all string decompositions

147. **Write a program which takes as input a log file and returns a pair of pages which have the highest affinity:** PL 13.14 Find a highest affinity pair

148. **You are given a sequence of users where each user has a unique 32-bit integer key and a set of attributes specified as strings. When you read a user, you should pair that user with another previously read user with identical attributes who is currently unpaired, if such a user exists. If the user cannot be paired, you should add him to the unpaired set.:** PL 13.15 Pair users by attributes

149. **Test the Collatz conjecture for the first n positive integers:** PL 13.16 Test the Collatz conjecture

150. **Design a hash function for chess game states. Your function should take a state and the hash code for that state, and a move, and efficiently compute the hash code for the updated state.:** PL 13.17 Implement a hash function for chess

151. **Write a program that takes as input a string and a set of strings, and returns the shortest prefix of the string which is not a prefix of any string in the set.:** PL 13.18 Find the shortest unique prefix

152. **Write a program which takes an input two sorted arrays, and returns a new array containing elements that are present in both of the input arrays. The input arrays may have duplicate entries, but the returned array should be free of duplicates.**
**Input: <2,3,3,5,5,6,7,7,8,12>, <5,5,6,8,8,9,10,10>**
**Output: <5,6,8>:** PL 14.1 Compute the intersection of two sorted arrays

153. **Write a program which takes an input two sorted arrays of integers, and updates the first to the combined entries of the two arrays in sorted order. Assume the first array has enough empty entries at its end to hold the result.:** PL 14.2 Implement mergesort in-place

154. **Given a string, print in alphabetical order each character that appears in the string, and the number of times that it appears.**
**Input: bcdacebe**
**Output: (a,1), (b,2), (c,2), (d,1), (e,2):** PL 14.3 Count the frequencies of characters in a sentence

155. **Design an efficient algorithm for removing all the duplicates from an array.:** PL 14.4 Find unique elements

156. **Write a program that takes a set of events, and determines the maximum number of events that take place concurrently.:** PL 14.5 Render a calendar

157. **Write a program which takes as input an array of disjoint closed intervals with integer endpoints, sorted by increasing order of left endpoint, and an interval to be added, and returns the union of the intervals in the array and the added interval. Your result should be expressed as a union of disjoin intervals sorted by left endpoint.:** PL 14.6 Sets of disjoint intervals

158. **Design an algorithm that takes as input a set of intervals, and outputs their union expressed as a set of disjoint intervals:** PL 14.7 Compute the union of intervals

159. **You are given a set of closed intervals. Design an efficient algorithm for finding a minimum number of numbers that cover all the intervals:** PL 14.8 The interval covering problem

160. **Design an algorithm that takes as input an array of numbers, which are the durations of the tasks, and computes a set of pairs such that the maximum pair sum is minimum. Each task must be in exactly one pair.:** PL 14.9 Compute an optimum assignment of tasks

161. **You are given an array of objects. Each object has a field that is to be treated as a key. Rearrange the elements of the array so that objects with equal keys appear together. The order in which distinct keys appear is not important:** PL 14.10 Implement counting sort

162. **Design an algorithm that takes as input two teams and the heights of the players in the teams and checks if it is possible to place players to take the photo subject to the placement constraint.:** PL 14.11 Team photo day- 1

163. **Implement a routine which sorts lists efficiently. It should be a stable sort, i.e., the relative positions of equal elements must remain unchanged.:** PL 14.12 Implement a fast sorting algorithm for lists

164. **Write a program which takes an array of positive integers and returns the smallest number which is not to the sum of a subset of elements of the array.:** PL 14.13 Compute the smallest non-constructible change

165. **Design an algorithm for computing the salary cap, given existing salaries and the target payroll.:** PL 14.14 Compute a salary threshold

166. **A text file is read into an array of characters with null characters indicating the line boundaries. Write a program that takes as input such as array, and prints the lines in sorted order. The array corresponds to a large number of lines, and while lines most are short, some are very long.:** PL 14.15 Implement a variable-length sort

167. **What is the minimum number of five man time-trials needed to determine the top three cyclists from a set of 25 cyclists?:** PL 14.16 Schedule time trials

168. **How would you organize a tournament with 128 players to minimize the number of matches needed to find the best player? How many matches do you need to find the best and the second best player?:** PL 14.17 Find the winner and runner-up

169. **Write a program that takes as input a binary tree and checks if the tree satisfies the BST property:** PL 15.1 Test if a binary tree satisfies the BST property

170. **Write a program that takes as input a BST and a value, and returns the node whose key equals the input value and appears first in an inorder traversal of the BST.:** PL 15.2 Find the first occurrence of a key in a BST

171. **Write a program that takes as input BST and a value, and returns the first key that would appear in an inorder traversal which is greater than the input value.:** PL 15.3 Find the first key larger than a given value in a BST

172. **Write a program that takes as input a BST and an integer k, and returns the k largest elements in the BST in decreasing order.:** PL 15.4 Find the k largest elements in a BST

173. **Design an algorithm that takes as input a BST and two nodes, and returns the LCA of the two nodes.:** PL 15.5 Compute the LCA in a BST

174. **Suppose you are given the sequence in which keys are visited in an inorder traversal of a BST, and all keys are distinct. Can you reconstruct the BST from this sequence? If so, write a program to do so. Solve the same problem for preorder and postorder traversal sequences.:** PL 15.6 Reconstruct a BST from traversal data

175. **Design an algorithm that takes three sorted arrays and returns one entry from each such that the minimum interval containing these three entries is as small as possible.:** PL 15.7 Find the closest entries in three sorted arrays

176. **Write a function to read a log file line, and a function to find the k most visited pages. Optimize performance for the situation where calls to the two functions are interleaved.:** PL 15.8 The most visited pages problem

177. **Write a function to read a log file line, and a function to find the k most visited pages. Optimize performance for the situation where calls to the two functions are interleaved. Also, restrict the k most visited pages to the pages whose visit-time is no less than W before the most recent page's visit-time. Here W, the window size, is an input parameter fixed for a given log file. You can assume visit-times increase as you process the file. Minimize memory usage.:** PL 15.9 Find the most visited pages in a window

178. **How would you build a BST of minimum possible height from a sorted array?:** PL 15.10 Build a BST from a sorted array

179. **Write a program that takes as input a doubly linked list of sorted numbers and builds a height-balanced BST on the entries in the list. You cannot use dynamic memory allocation - reuse the nodes of the list for the BST, using the previous and next fields for the left and right children, respectively:** PL 15.11 Convert a sorted doubly linked list into a BST

180. **Design an algorithm that takes as input a BST and returns a sorted doubly linked list on the same elements. Your algorithm should not do any dynamic allocation. The original BST does not have to be preserved; use its nodes as the nodes of the resulting list.:** PL 15.12 Convert a BST to a sorted doubly linked list

181. **Design an algorithm that takes as input two BSTs and merges them to form a balanced BST. You cannot use dynamic allocation. For any node, you can update its left and right subtree fields, but cannot change its key.:** PL 15.13 Merge two BSTs

182. **Design efficient functions for inserting and removing keys in a BST. Assume that all elements in the BST are unique, and that your insertion method must preserve this property. You cannot change the contents of any node.:** PL 15.14 Insertion and deletion in a BST

183. **Write a program which takes two nodes in a BST and a third node, the "middle" node, and determines if one of the two nodes is an ancestor and the other a descendent of the middle.:** PL 15.15 Test if three BST nodes are totally ordered

184. **Write a program that takes as input a binary tree with integer keys, and determines if it is an almost BST. If it is an almost BST, reconstruct the corresponding BST.:** PL 15.16 Test if a binary tree is an almost BST

185. **Write a program which takes as input a file containing test scores and returns the student who has the maximum score averaged across his or her top three tests. If the student has fewer than three test scores, ignore the student.:** PL 15.17 Compute the average of the top three scores

186. **Write a program that takes as input a BST and an interval and returns the BST keys that lie in the interval.:** PL 15.18 The range lookup problem

187. **Write a program that computes the view from above. The input is a sequence of line segments, and the output should be in the same format. You can assume no two segments whose intervals overlap have the same height.:** PL 15.19 The view from above

188. **Write a program that takes a min-first BST and a value as input, and checks whether the tree contains the value:** PL 15.20 Searching a min-first BST

189. **Design a data structure that implements the following methods:**
**- insert: add a client with specified credit, replacing any existing entry for the client.**
**- remove: delete the specified client**
**- lookup: return the number of credits associated with the specified client.**
**- add-to-all: increment the credit count for all current clients by the specified amount**
**- max: return a client with the highest number of credits:** PL 15.21 Add credits

190. **Suppose each node in a BST has a size field, which denotes the number of nodes at the subtree rooted at that node, inclusive of the node. How would you efficiently compute the number of keys that lie in a given interval? Your solution should work in the presence of duplicate keys.:** PL 15.22 Count the number of entries in an interval

191. **Exactly n rings on P1 need to be transferred to P2, possibly using P3 as an intermediate, subject to the stacking constraint. Write a program that prints a sequence of operations that transfers all the rings from P1 to P2.:** PL 16.1 The towers of Hanoi problem
1. Recursion

192. **Design an algorithm that takes a string s and a string r, assumed to be a well-formed ESRE, and checks if r matches s.:** PL 16.2 Implement regular expression matching
1. Recursion

193. **Write a program which takes as input a positive integer n, and returns a list of all distinct nonattacking placements of n queens on an n*n chessboard.:** PL 16.3 Enumerate all non-attacking placements of N-Queens
1. Recursion

194. **Write a program that takes as input an array A of n distinct integers and prints all permutations of A. No permutation appears more than once. Your program should use O(n) space.:** Pl 16.4 Enumerate permutations
1. Recursion

195. **Implement a method that takes as input a set S of n distinct elements, and returns the power set of S.:** PL 16.5 Enumerate the power set
1. Recursion

196. **Write a program which takes as input integers n and k, and computes all subsets of size k of the set (1,2, ..., n):** PL 16.6 Enumerate all subsets of size k
1. Recursion
2. Bits (iteration with same # of 1's)

197. **Write a program that takes as input a positive integer k and returns a list of all the strings that have k pairs of matched parens.:** PL 16.7 Enumerate strings of balanced parens
1. Recursion

198. **Compute all palindromic decompositions of a given string.**

**Input: "0204451881"**
**Output: {"020", "44", "5", "1881"},...:** PL 16.8 Enumerate palindromic decompositions
1. Recursion

199. **Write a program which takes as input an integer n and returns all distinct binary trees with n nodes.:** PL 16.9 Enumerate binary trees
1. Recursion

200. **Implement a sudoku solver:** PL 16.10 Implement a sudoku solver
1. Recursion
check before setting the candidate

201. **Write a program that takes as input a single positive integer n and returns an n-bit Gray code.:** PL 16.11 Compute a gray code
1. Recursion

202. **Given an array of digits A and a nonnegative integer k, intersperse multiplies (x) and adds(+) with the digits of A such that the resulting arithmetical expression evaluates to k.**

**input: <1,2,3,2,5,3,7,8,5,9>, k=995**
**output: 123 + 2 + 5 x 3 x 7 + 85 x 9:** PL 16.12 Synthesize an expression
1. Recursion
iterate the terms to calculate the multiplication first then sum the rest

203. **Design an efficient algorithm that takes an array A of n numbers and returns the number of inverted pairs of indices.:** PL 16.13 Count inversions
1. O(n^2) brute force
2. O(nlogn) Divide and conquer (merge sort)
inversions += m - l

204. **Design an efficient algorithm to compute the diameter of a tree:** PL 16.14 Compute the diameter of a tree
Divide and conquer

205. **Design an efficient algorithm for computing the skyline.:** PL 16.15 Draw the skyline
Divide and conquer

206. **You are given a list of pairs of points in the 2D Cartesian plane. Each point has integer X and Y coordinates. How would you find the two closet points?:** PL 16.16 Find the two closest points
Divide and conquer

207. **You have an aggregate score s and W which specifies the points that can be scored in an individual play. How would you find the number of combinations of plays that result in an aggregate score of s? How would you compute the number of distinct sequences of individual plays that result in a score of s?:** PL 17.1 Count the number of score combinations
Dynamic programming
- combination: 1. score 2. item
- permutation: 1. item 2. score

208. **Given two strings, represented as arrays of characters A and B, compute the minimum numbers of edits needed to transform the first string into the second string.:** PL 17.2 Compute the Levenshtein distance
Dynamic programming
- A[i]!=B[j] -> 1 + min(T[i-1,j-1], T[i][j-1], T[i-1][j])
- A[i] =B[j] -> T[i-1][j-1]

209. **Design an efficient algorithm for computing nCk which has the property that it never overflows if nCk can be represented as a 32-bit integer; assume n and k are integers.:** PL 17.3 Compute the binomial coefficients
Dynamic programming
- C(n,k) = C(n-1,k-1) + C(n-1,k)

210. **How many ways can you go from the top-left to the bottom-right in an n x m 2D array? How would you count the number of ways in the presence of obstacles, specified by an n x m Boolean 2D array B, where a true represents an obstacle.:** PL 17.4 Count the number of ways to traverse a 2D array
Dynamic programming

211. **Write a program that computes the maximum value of fish a fisherman can catch on a path from the upper leftmost point to the lower rightmost point. The fisherman can only move down or right.:** PL 17.5 Plan a fishing trip.
Dynamic programming

212. **Design an algorithm that takes as arguments a 2D array A and a 1D array S, and determines whether S appears in A. If S appears in A, print the sequence of entries where it appears.:** PL 17.6 Search for a sequence in a 2D array
- backtracking+memorization
- hash with (i,j,length)

213. **Design an algorithm for the knapsack problem that selects a subset of items that has maximum value and weights at most w ounces. All items have integer weights and values:**
PL 17.7 The knapsack problem
- max(V[i-1][w], V[i-1][w-w[i]] + v[i])
- SC: O(nw) -> O(w)

214. **Write a program that determines a sequence of steps by which the required amount of milk can be obtained using the worn-out jugs. The milk is being added to a large mixing bowl, and hence cannot be removed from the bowl. Furthermore, it is not possible to pour one jug's contents into another. Your scheme should always work, i.e., return between 2100 and 2300 mL of milk, independent of how much is chosen in each individual step, as long as that quantity satisfies the given constraints.:** PL 17.8 Measure with defective jugs
- skipped

215. **How would you programmatically determine if a tie is possible in a presidential election with two candidates?:** PL 17.9 Test if a tie is possible
- find the subset sum where total = sum / 2

216. **Two thieves have successfully completed a burglary. They want to know how to divide the stolen items into two groups such that the difference between the value of these two groups is minimized.**

**Let array A be an array of n positive integers. Entry A[i] is the value of the i-th stolen item. Design an algorithm that computes a subset such that the difference is minimized.:**
PL 17.10 Divide the spoils fairly
```
for item in A:
for s in sum/2..item
if !bitset.get(s-item): bitset.set(s)

for s in sum/2..0
if bitset.get(s): return sum-i-i

return sum
```

217. **Given a circular array A, compute its maximum subarray sum in O(n) time, where n is the length of A. Can you devise an algorithm that takes O(n) time and O(1) space?:** PL 17.11
Compute the maximum subarray sum in a circular array
```
max(
maxSum(A, max),
total-maxSum(A, min)
)
```

218. **Given a dictionary, i.e., a set of strings, and a string s, design an efficient algorithm that checks whether s is the connotation of a sequence of dictionary words. If such a concatenation exists, your algorithm should output it.:** PL 17.12 The bedbathandbeyond.com problem
```
T[i] = j if (D.contains(s.substring(i,j)) && T[j+1]))
```

219. **Given c cases and d drops, what is the maximum number of floors that you can test in the worst-case?:** PL 17.13
Determine the critical height
```
F(c,d) = F(c-1,d-1)+1+F(c,d-1)

if (c == 0 || d == 0) return 0;
else if (c == 1) return d;
else
if (F[c][d] == -1)
F[c][d] = getHeight(c,d-1,F) + getHeight(c-1,d-1,F)+1;

return F[c][d]
```

220. **Write a program that takes as input a sequence of integer-valued arrays, <A0,A1,A2,…,An-1> and returns the minimum weight path from the top number to a number in Row n.:**
PL 17.14 Find the maximum(minimum) weight path in a triangle
- minimum weight for the bottom row
- move up, recycle the row

221. **Design an efficient algorithm for computing the maximum margin of victory for the starting player in the pick-up-coins game.:** PL 17.15 Pick up coins for maximum gain
```
F(a,b) = max(
C[a] + min(f(a+2,b), f(a+1,b-1))
C[b] + min(f(a+1, b-1), f(a,b-2))
)
```

222. **How would you compute a palindrome decomposition of a string s that uses a minimum number of substrings?:** PL 17.16 Decompose into palindrome strings
```
for i in s.length()-1 .. 0:
for j in i .. s.length()-1:
if (s[i]==s[j] && (j - i < 2 || T[i+1][j-1])) {
T[i][j]=true
T[i]=min(T[i],1+T[j+1])
}
return T[0]
```

223. **Design an algorithm that takes as input strings s1, s2, and t, and determines if t is an interleaving of s1 and s2.:** PL 17.17
Test if s is an interleaving of s1 and s2
Dynamic programming

224. **Write a program which takes as inputs n and k returns the number of ways in which you can advance to the destination**

**Input: n=4, k=2**
**Output: <1,1,1,1>, <1,1,2>, <1,2,1>, <2,1,1>, <2,2>:** PL 17.18 Count the number of steps in a board game
Dynamic programming

225. **Assuming elections are statistically independent and that the probability of a Republican winning Election i is pi, how would you compute the probability of a Republican majority?:** PL 17.19 Compute the probability of a Republican majority
Dynamic programming

226. **Given text, i.e., a string of words separated by single blanks, decompose the text into lines such that no word is split across lines and the messiness of the decomposition is minimized. Each line can hold no more than L characters. How would you change your algorithm if the messiness i the sum of the messinesses of all but the last line?:** PL 17.20 The pretty printing problem
Dynamic programming

227. **Given an array A of n numbers, find a longest subsequence <i0,...,ik-1> such that ij < ij+1 and A[ij] <= A[ij+1] for any j in [0, k-2]**

**Input: <0, 8, 4, 12, 2, 10, 6, 14, 1, 9>**
**Output: <0, 4, 10, 14> or <0, 2, 6, 9>:** PL 17.21 Find the longest nondecreasing subsequence
1. O(n^2) dynamic programming
2. O(nlogn) greedy algorithm + binary search
X[M[mid]] < X[i]
M[lo] = i

228. **Design an algorithm for minimizing power that takes as input a rooted tree and assigns each node to a low or high voltage, subject to the design constraint (low voltage -> high voltage):** PL 17.22 Voltage selection in a logic circuit
Dynamic programming

229. **Let A be an n x m Boolean 2D array. Design an efficient algorithms for the following two problems:**

**1. What is the largest 2D subarray containing only 1s?**
**2. What is the largest square 2D subarray containing only 1s?**

**What are the time and space complexities of your algorithms as a function of n and m?:** PL 17.23 Find the maximum 2D subarray
Dynamic programming

230. **Given a set of symbols with corresponding frequencies, find a code book that has the smallest average code length.:** PL 18.1 Implement Huffman Coding
1. Greedy Algorithms and Invariants

231. **Given n queries, compute an order in which to process queries that minimizes the total waiting time.:** PL 18.2 Implement a schedule which minimizes waiting time
1. Greedy Algorithms and Invariants

232. **A one-dimensional container is specified by an array of n nonnegative integers, specifying the height of each unit-width rectangle. Design an algorithm for computing the capacity of the container.:** PL 18.3 Trapping water
1. Greedy Algorithms and Invariants

233. **You have n users with unique hash codes h(0) through h(n-1), and m servers. The hash codes are ordered by index. User i requires b(i) bytes of storage. The values k(0) < K(1) < k(m-2) are used to assign users to servers. Specifically, the user with hash code c gets assigned to the server with the lowest ID l such that c <= k(i), or to Server m - 1 if no such i exists. This restricted mapping of users to servers means that the user-to-server lookup can be implemented with a BST on m - 1 nodes, rather than a hash table on n users. The load on a server is the sum of the bytes of storage of all users assigned to that server. Compute values for k(0), k(1), ..., k(m-2) that minimizes the load on the most heavily loaded server.:** PL 18.4 Load balancing
1. DP O((n^2)m)
2. Greedy O(nlogW) where W = sum(bi)
- minimizing around the difference from mean doesn't work

234. **Implement first-fit to run in O(nlogn) time:** PL 18.5 Pack for USPS Priority Mail
1. Naive O(n^2)
2. Tournament tree O(nlogn)

- ignore coordinates for items
- minimize # of non-zero weight leaves in the tournament tree (zero weight leaves are un-used boxes)

235. **Design an algorithm that takes as input an array A and a number t, and determines if A 3-creates t.:** PL 18.6 The 3-sum problem
1. Greedy Algorithms and Invariants

236. **Given an instance of the gasup problem, how would you efficiently compute an ample city, if one exists?:** PL 18.7 The gasup problem
1. Greedy Algorithms and Invariants

237. **Design an algorithm for efficiently computing the k smallest real numbers of the form a + b*sqrt(2) for nonnegative integers a and b.:** PL 18.8 Enumerate numbers of the form a + b*sqrt(2)
1. Brute force O(k^2)
2. BST O(klogk)
3. Greedy O(k)

238. **You are reading a sequence of words from a very long stream. You know a priori that more than half the words are repetitions of a single word w (the "majority element") but the positions where w occurs are unknown. Design an algorithm that makes a single pass over the stream and uses only a constant amount of memory to identify w.:** PL 18.9 Find the majority element
1. Greedy Algorithms and Invariants

239. **Design an algorithm that takes an abs-sorted array A and a number K, and returns a pair of indices of elements in A that sum up to K.**

**Input: <-49, 75, 103, -147, 164, -197, -238, 314, 348, -422>, K= 167**
**Output: (3, 7):** PL 18.10 Search for a pair-sum in an abs-sorted array
1. Three passes O(n)

240. **Define the water trapped by entries i and j in an integer array A to be |j-i| x min(A[i], A[j]). Write a program which takes as input A and returns the pairs of entries that trap the maximum amount of water.:** PL 18.11 Compute the maximum water trapped by a pair of vertical lines
1. Greedy Algorithms and Invariants

241. **You are reading a sequence of string separated by whitespace. You are allowed to read the sequence twice. Devise an algorithm that uses O(k) memory to identify the words that occur at least ceil(n/k) times, where n is the length of the sequence.:** PL 18.12 The heavy hitter problem
1. Greedy Algorithms and Invariants

242. **Design an algorithm that takes as input an array A of n numbers and a key k, and returns a longest subarray of A for which the subarray sum is less than or equal to k.:** PL 18.13 Find the longest subarray whose sum <= k
1. brute force T: O(n^3)
2. brute force + prefix sum T:O(n^2) S:O(n)
3. greedy + min possible sum T:O(n) S:O(n)

243. **Let A be an array of n numbers encoding the heights of adjacent buildings of unit width. Design an algorithm to compute the area of the largest rectangle contained in this skyline.:** PL 18.14 Compute the largest rectangle under the skyline
1. brute force T:O(n^3)
2. brute force T:O(n^2) S:(n)
3. greedy + stack T:O(n) S:(n)
- max rectangle in histogram + accumulate heights

244. **Given a 2D array of black and white entries representing a maze with designated entrance and exit points, find a path from the entrance to the exit, if one exists.:** PL 19.1 Search a maze
1. DFS O(|V| + |E|)
2. BFS for shortest path

245. **Implement a routine that takes a n x m Boolean array A together with an entry (x,y) and flips the color of the region associated with (x,y):** PL 19.2 Paint a Boolean Matrix
1. Graph

246. **Let A be a 2D array whose entries are either W or B. Write a program that takes A, and replaces All Ws that cannot reach the boundary with a B.:** PL 19.3 Compute enclosed regions
1. Graph

247. **Design an algorithm that takes a reference to a vertex u, and creates a copy of the graph on the vertices reachable from u. Return the copy of u.:** PL 19.4 Clone a graph
1. Graph

248. **Given a dictionary D and two strings s and t, write a program to determine if s produces t. Assume that all characters are lowercase alphabets. If s does produce t, output the length of a shortest production sequence; otherwise, output - 1.:** PL 19.5 Transform one string to another
1. Graph

249. **Design an algorithm that takes a set of pins and a set of wires connecting pairs of pins, and determines if it is possible to place some pins on the left half of a PCB, and the remainder on the right half, such that each wire is between left and right halves. Return such a division, if one exists.:** PL 19.6 Making wired connections
1. Graph

250. **Let G = (V,E) be a connected undirected graph. How would you efficiently check if G is 2E-connected? Can you make your algorithm run in O(|V|) time? How would you check if G is 2V-connected?:** PL 19.7 Test degrees of connectedness
1. Graph

251. **How would you generalize your solution to Problem 14.11 on Page 104 to determine the largest number of teams that can be photographed simultaneously subject to the same constraints?:** PL 19.8 Team photo day
1. Graph

252. **Given an instance of the task scheduling problem, compute the least amount of time in which all the tasks can be performed, assuming an unlimited number of servers. Explicitly check that the system is feasible.:** PL 19.9 Compute a minimum delay schedule, unlimited resources
1. Graph

253. **Design an algorithm which takes an input a graph G=(V,E), directed or undirected, a nonnegative cost function on E, and vertices s and t; your algorithm should output a path with the fewest edges amongst all shortest paths from s to t.:** PL 19.10 Compute a shortest path with fewest edges
1. Graph

254. **Devise an efficient algorithm which takes the existing highway network (specified as a set of highway sections between pairs of cities) and proposals for new highway sections, and returns a proposed highway section which minimizes the shortest driving distance between El Paso and Corpus Christi. All sections, existing and proposed, allow for bi-directional traffic.:** PL 19.11 Road network
1. Graph

255. **Design an efficient algorithm to determine whether there exists an arbitrage - a way to start with a single unit of some commodity C and convert it back to more than one unit of C through a sequence of exchanges.:** PL 19.12 Test if arbitrage is possible
1. Graph