|  |
| --- |
| 5.8 REVERSE DIGITS  Write a program which takes an integer and returns the integer corresponding to the digits of the input written in reverse order. For example, the reverse of 42 is 24, and the reverse of -314 is -413.  Hint: How would you solve the same problem if the input is presented as a string?  P54(P67) |
| 6.2 INCREMENT AN ARBITRARY-PRECISION INTEGER  Write a program which takes as input an array of digits encoding a decimal number D and updates the array to represent the number D + 1. For example, if the input is (1,2,9) then you should update the array to (1,3,0). Your algorithm should work even if it is implemented in a language that has finite-precision arithmetic.  Hint: Experiment with concrete examples.  P65(P78) |
| 6.12 SAMPLE ONLINE DATA  This problem is motivated by the design of a packet sniffer that provides a uniform sample of packets for a network session.  Design a program that takes as input a size k, and reads packets, continuously maintaining a uniform random subset of size k of the read packets.  Hint: Suppose you have a procedure which selects k packets from the first n > k packets as specified. How would you deal with the (n + l)th packet?  P79(P92) |
| 7.5 TEST PALINDROMICITY  For the purpose of this problem, define a palindromic string to be a string which when the entire non-alphanumeric are removed it reads the same front to back ignoring case.  For example, "A man, a plan, a canal, Panama." and "Able was I, ere I saw Elba!" are palindromic, but "Ray a Ray" is not.  Implement a function which takes as input a string s and returns true if s is a palindromic string.  Hint: Use two indices.  P100(P113) |
| 7.6 REVERSE ALL THE WORDS IN A SENTENCE  Given a string containing a set of words separated by whitespace, we would like to transform it to a string in which the words appear in the reverse order. For example,  "Alice likes Bob" transforms to "Bob likes Alice". We do not need to keep the original string.  Implement a function for reversing the words in a string s.  Hint: It's difficult to solve this with one pass.  P101(P114) |
| 8.4 TEST FOR OVERLAPPING LISTS—LISTS ARE CYCLE-FREE  Given two singly linked lists there may be list nodes that are common to both. (This may not be a bug—it may be desirable from the perspective of reducing memory footprint, as in the flyweight pattern, or maintaining a canonical form.) For example, the lists in Figure 8.6 overlap at Node I.    Figure 8.6: Example of overlapping lists.  Write a program that takes two cycle-free singly linked lists, and determines if there exists a node that is common to both lists.  Hint: Solve the simple cases first.  P119(P132) |
| 8.7 REMOVE THE kTH LAST ELEMENT FROM A LIST  Without knowing the length of a linked list, it is not trivial to delete the kth last element in a singly linked list.  Given a singly linked list and an integer k, write a program to remove the kth 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.  Hint: If you know the length of the list, can you find the kth last node using two iterators?  P123(P136) |
| 9.2 EVALUATE RPN EXPRESSIONS  A string is said to be an arithmetical expression in Reverse Polish notation (RPN) if:  (1.) It is a single digit or a sequence of digits, prefixed with an option -, e.g., "6", "123", "-42".  (2.) It is of the form “A,B,o" where A and B are RPN expressions and o is one of +/-/X, /•  For example, the following strings satisfy these rules: "1729", "3,4, +,2, X,1, +",  "1,1, +,-2, x", "-641,6, /, 28, /".  An RPN expression can be evaluated uniquely to an integer, which is determined recursively. The base case corresponds to  Rule (1.), which is an integer expressed in base-10 positional system.  Rule (2.) corresponds to the recursive case, and the RPNs are evaluated in the natural way, e.g., if A evaluates to 2 and B evaluates to 3, then "A,B,x" evaluates to 6.  Write a program that takes an arithmetical expression in RPN and returns the number that the expression evaluates to.  Hint: Process subexpressions, keeping values in a stack. How should operators be handled?  P135(P148) |
| 9.8 IMPLEMENT A CIRCULAR QUEUE  A queue can be implemented using an array and two additional fields, the beginning and the end indices. This structure is sometimes referred to as a circular queue.  Both enqueue and dequeue have 0(1) time complexity. If the array is fixed, there are a maximum number of entries that can be stored. If the array is dynamically resized, the total time for m combined enqueue and dequeue operations is 0(m).  Implement a queue API using an array for storing elements. Your API should include a constructor function, which takes as argument the initial capacity of the queue, enqueue and dequeuer functions, and a function which returns the number of elements stored. Implement dynamic resizing to support storing an arbitrarily large number of elements.  Hint: Track the head and tail. How can you differentiate a full queue from an empty one?  P145(P158) |
| 10.2 TEST IF A BINARY TREE IS SYMMETRIC  A binary tree is symmetric if you can draw a vertical line through the root and then the left subtree is the mirror image of the right subtree. The concept of a symmetric binary tree is illustrated in Figure 10.3 on the facing page.    Write a program that checks whether a binary tree is symmetric.  Hint: The definition of symmetry is recursive.  P154(P167) |
| 10.12 RECONSTRUCT A BINARY TREE FROM TRAVERSAL DATA  Many different binary trees yield the same sequence of keys in an inorder, preorder, or postorder traversal. However, given an **inorder** traversal and one of any two other traversal orders of a binary tree, there exists a unique binary tree that yields those orders, assuming each node holds a distinct key.  For example, the unique binary tree whose inorder traversal sequence is (F,B,A,E,H,C,D,I,G) and whose preorder traversal sequence is (H,B,F,E,A,C,D,G,I) is given in Figure 10.5 on the following page.    Figure 10.5: A binary tree—edges that do not terminate in nodes denote empty subtrees.  Given an inorder traversal sequence and a preorder traversal sequence of a binary tree write a program to reconstruct the tree. Assume each node has a unique key.  Hint: Focus on the root.  P165(P178) |
| 11.3 SORT AN ALMOST-SORTED ARRAY  Often data is almost-sorted—for example, a server receives timestamped stock quotes and earlier quotes may arrive slightly after later quotes because of differences in server loads and network routes. In this problem we address efficient ways to sort such data.  Write a program which takes as input a very long sequence of numbers and prints the numbers in sorted order. Each number is at most k away from its correctly sorted position. (Such an array is sometimes referred to as being For example, no number in the sequence (3,-1,2,6,4,5,8} is more than 2 away from its final sorted position.  Hint: How many numbers  P180(P193) |
| 12.3 SEARCH A CYCLICALLY SORTED ARRAY  An array is said to be cyclically sorted if it is possible to cyclically shift its entries so that it becomes sorted. For example, the array in Figure 12.2 on the facing page is cyclically sorted—a cyclic left shift by 4 leads to a sorted array.  Design an O(log n) algorithm for finding the position of the smallest element in a cyclically sorted array. Assume all elements are distinct. For example, for the array in Figure 12.2 on the next page, your algorithm should return 4.  Hint: Use the divide and conquer principle.  378 478 550 631 103 203 220 234 279 368  A[0] A[1] A[2] A[3] A[4] A[5] A[6] A[7] A[8] A[9]  Figure 12.2: A cyclically sorted array.  P192(P205) |
| 12.9 FIND THE MISSING IP ADDRESS  The storage capacity of hard drives dwarfs that of RAM. This can lead to interesting space-time trade-offs.  Suppose you were given a file containing roughly one billion IP addresses, each of which is a 32-bit quantity. 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.  Hint: Can you be sure there is an address which is not in the file?  P202(P215) |
| 13.1 TEST FOR PALINDROMIC PERMUTATIONS  A palindrome is a string that reads the same forwards and backwards, e.g., "level", "rotator", and "foobaraboof".  Write a program to test whether the letters forming a string can be permuted to form a palindrome. For example, "edified" can be permuted to form "deified".  Hint: Find a simple characterization of strings that can be permuted to form a palindrome.  P212(P225) |
| 13.6 FIND THE NEAREST REPEATED ENTRIES IN AN ARRAY  People do not like reading text in which a word is used multiple times in a short paragraph. You are to write a program which helps identify such a problem.  Write a program which takes as input an array and finds the distance between a closest pair of equal entries. For example,  if s = ("All", "work", "and", "no", "play", "makes", "for", "no", "work", "no", "fun", "and", "no", "results"), then the second and third occurrences of "no" is the closest pair.  Hint: Each entry in the array is a candidate.  P218(P231) |
| 14.4 RENDER A CALENDAR  Consider the problem of designing an online calendaring application. One component of the design is to render the calendar, i.e., display it visually.  Suppose each day consists of a number of events, where an event is specified as a start time and a finish time. Individual events for a day are to be rendered as non-overlapping rectangular regions whose sides are parallel to the X- and Y-axes.  Let the X-axis correspond to time. If an event starts at time b and ends at time e, the upper and lower sides of its corresponding rectangle must be at b and e, respectively.  Figure 14.1 represents a set of events.  Suppose the Y-coordinates for each day's events must lie between 0 and L (a pre-specified constant), and each event's rectangle must have the same "height" (distance between the sides parallel to the X-axis). Your task is to compute the maximum height an event rectangle can have. In essence, this is equivalent to the following problem.    Write a program that takes a set of events, and determines the maximum number of events that take place concurrently.  Hint: Focus on endpoints.  P240(P253) |
| 15.4 COMPUTE THE LCA IN A BST  Since a BST is a specialized binary tree, the notion of lowest common ancestor, as expressed in Problem 10.4 on Page 157, holds for BSTs too.  In general, computing the LCA of two nodes in a BST is no easier than computing the LCA in a binary tree, since structurally a binary tree can be viewed as a BST where all the keys are equal. However, when the keys are distinct, it is possible to improve on the LCA algorithms for binary trees.  Design an algorithm that takes as input a BST and two nodes, and returns the LCA of the two nodes. For example, for the BST in Figure 15.1 on Page 255, and nodes C and G, your algorithm should return B. Assume all keys are distinct. Nodes do not have references to their parents.  Hint: Take advantage of the BST property.  P261(P274) |
| 15.8 THE MOST VISITED PAGES PROBLEM  You are given a server log file containing billions of lines. Each line contains a number of fields. For this problem, the relevant field is an id denoting the page that was accessed.  Write a function to read the next line from a log file, and a function to find the k most visited pages, where k is an input to the function. Optimize performance for the situation where calls to the two functions are interleaved. You can assume the set of distinct pages is small enough to fit in RAM.  As a concrete example, suppose the log file ids appear in the following order:  g,a,t,t,a,a,a,g,t,c,t,a,t, i.e., there are four pages with ids a,c,g,t. After the first 10 lines have been read, the most common page is a with a count of 4, and the next most common page is t with a count of 3.  Hint: For each page, count of the number of times it has been visited.  P269(P282) |
| 16.3 GENERATE PERMUTATIONS  This problem is concerned with computing all permutations of an array. For example,  if the array is <2,3,5,7>  one output could be:  <2,3,5,7>, <2,3,7,5>, <2,5,3,7>, <2,5,7,3>,  <2,7,3,5>, <2,7,5,3>, <3,2,5,7>, <3,2,7,5>,  <3,5,2,7>, <3,5,7,2>, <3,7,2,5>, <3,7,5,2>,  <5,2,3,7>, <5,2,7,3>, <5,3,2,7>, <5,3,7,2>,  <5,7,2,3>, <5,7,2,3>, <7,2,3,5>, <7,2,5,3>,  <7,3,2,5>, <7,3,5,2>, <7,5,2,3>, <7,5,3,2>.  (Any other ordering is acceptable too.)  Write a program which takes as input an array of distinct integers and generates all permutations of that array. No permutation of the array may appear more than once.  Hint: How many possible values are there for the first element?  P287(P300) |
| **17.3 COUNT THE NUMBER OF WAYS TO TRAVERSE A 2D ARRAY**  In this problem you are to count the number of ways of starting at the top-left comer of a 2D array and getting to the bottom-right comer. All moves must either go right or down.  For example, we show three ways in a 5 X 5 2D array in Figure 17.5. (As we will see, there are a total of 70 possible ways for this example.)    Figure 17.5: Paths through a 2D array.  Write a program that counts how many ways you can go from the top-left to the bottom-right in a 2D array.  Hint: If i > 0 and j > 0, you can get to (i, j) from (i-1, j) or (j-1, i).  Solution: A brute-force approach is to enumerate all possible paths  P312(P325) |
| **17.6 THE KNAPSACK PROBLEM**  A thief breaks into a clock store. Each clock has a weight and a value, which are known to the thief. His knapsack cannot hold more than a specified combined weight. His intention is to take clocks whose total value is maximum subject to the knapsack's weight constraint.  His problem is illustrated in Figure 17.8 on the next page.    If the knapsack can hold at most 130 ounces, he cannot take all the clocks. If he greedily chooses clocks, in decreasing order of value-to-weight ratio, he will choose P, H,O, B, I, and L in that order for a total value of $669. However, {H,J,O} is the optimum selection, yielding a total value of $695.  Write a program for the knapsack problem that selects a subset of items that has maximum value and satisfies the weight constraint. All items have integer weights and values. Return the value of the subset.  Hint: Greedy approaches are doomed.  P317(P330) |
| 18.5 FIND THE MAJORITY ELEMENT  Several applications require identification of elements in a sequence which occur more than a specified fraction of the total number of elements in the sequence. For example, we may want to identify the users using excessive network bandwidth or IP addresses originating the most Hypertext Transfer Protocol (HTTP) requests. Here we consider a simplified version of this problem.  You are reading a sequence of strings. You know a priori that more than half the strings are repetitions of a single string (the "majority element") but the positions where the majority element occurs are unknown. Write a program that makes a single pass over the sequence and identifies the majority element. For example, if the input is (b,a,c,a,a,b,a,a,c,a), then a is the majority element (it appears in 6 out of the 10 places).  Hint: Take advantage of the existence of a majority element to perform elimination.  P341(P354) |
| 18.7 COMPUTE THE MAXIMUM WATER TRAPPED BY A PAIR OF VERTICAL LINES  An array of integers naturally defines a set of lines parallel to the Y-axis, starting from  x = 0 as illustrated in Figure 18.4(a). The goal of this problem is to find the pair of lines that together with the X-axis "trap" the most water. See Figure 18.4(b) for an example.    Figure 18.4: Example of maximum water trapped by a pair of vertical lines.  Write a program which takes as input an integer array and returns the pair of entries that trap the maximum amount of water.  Hint: Start with 0 and n - 1 and work your way in.  P345(P358) |
| **19.2 PAINT A BOOLEAN MATRIX**  Let A be a Boolean 2Darray encoding a black-and-white image. The entry A(a,b) can be viewed as encoding the color at entry (a,b). Call two entries adjacent if one is to the left, right, above or below the other. Note that the definition implies that an entry can be adjacent to at most four other entries, and that adjacency is symmetric, i.e., if eO is adjacent to entry el, then el is adjacent to eO.  Define a path from entry eO to entry el to be a sequence of adjacent entries, starting at eO, ending at el, with successive entries being adjacent. Define the region associated with  a point (i, j) to be all points (i', j') such that there exists a path from (i, j) to (i', j') in which all entries are the same color. In particular this implies (i, j) and (i', j') must be the same color.    Figure 19.6: The color of all squares associated with the first square marked with a x in (a) have been recolored to yield the coloring in (b). The same process yields the coloring in (c).  Implement a routine that takes an n X m Boolean array A together with an entry (x, y) and flips the color of the region associated with (x, y). See Figure 19.6 for an example of flipping.  Hint: Solve this conceptually, and then think about implementation optimizations.  P357(P370) |
| 20.8 IMPLEMENT A TIMER CLASS  Consider a web-based calendar in which the server hosting the calendar has to perform a task when the next calendar event takes place. (The task could be sending an email or a Short Message Service (SMS).) Your job is to design a facility that manages the execution of such tasks.  Develop a timer class that manages the execution of deferred tasks. The timer constructor takes as its argument an object which includes a run method and a string-valued name field. The class must support—  (1.) Starting a thread, identified by name, at a given time in the future; and  (2.) Canceling a thread, identified by name (the cancel request is to be ignored if the thread has already started).  Hint: There are two aspects—data structure design and concurrency.  P385(P398) |
| 21.16 DESIGN A RECOMMENDATION SYSTEM  Jingle wants to generate more page views on its news site. A product manager has the idea to add to each article a sidebar of clickable snippets from articles that are likely to be of interest to someone reading the current article.  Design a system that automatically generates a sidebar of related articles.  Hint: This problem can be solved with various degrees of algorithmic sophistication: none at all, simple frequency analysis, or machine learning.  P405(P418) |