Questionnaire

*Draft 0.2*

*Priyesh Wadhwa*

**Table of Contents**

1. Arrays and Strings 27

1.1. Find the elements common to two integer arrays 27

1.2. Given an array of size n. It contains numbers in the range 1 to n. Each number is present at least once except for 2 numbers. Find the missing numbers. 28

1.3. Given an array of size n, containing every element from 1 to n+1, except one. Find the missing element. 28

1.4. Given an array of size n. It contains numbers in the range 1 to n. Each number is present at least once except for 1 number. Find the missing number. 28

1.5. Find a missing and repeating integer from an array of N integers ranging from 1 to N. 28

1.6. You have an array in which every number is repeated odd number of times except one, you have to give that with O(n) time complexity 29

1.7. Find the two numbers with odd occurrences in an unsorted array 29

1.8. Suppose we have an array like 1,2,3,4,5,a,b,c,d,e where we have always even number of elements. First half of the elements are integers and second half are alphabets we have to change it to like 1,a,2,b,3,c,4,d,5,e in place i.e no use of any extra space, variables are allowed .. 31

1.9. Write a function that given two strings s1 and s2, it will return s1 minus s2, and it has to use same s1 location. ex: s1 = "abcdB" ,s2 = "b" ,s1 Minus s2 = "acdB" 31

1.10. Given an crypted array obtain the original text . Should be implemented without extra space . Eg: Crypt array : a3b4c3 decrypt array : aaabbbbccc Imagine the array contains sufficient memory to hold the decrypt ? 32

1.11. Given a string, compress using run-length encoding. Example: Input: aaabbac Output: a3b2ac 32

1.12. In a nxn matrix, data provided like below. We need to find the groups of 1s with the adjactent column and row. 33

1.13. Write an algorithm to find an element in a 2D array such that the element should be maximum in the row and minimum in the column? 34

1.14. How to rotate the array with o(n) or o(nlogn). eg) A[]={A,B,C,D,E} rotate Index – 2 It should be {C,D,E,A,B} 35

1.15. Given an array and a number K. You have to find out longest subset from the array whose all pair sum will be greater than k. ex: {8,3,4,1,6,2,5,7,9} and K=12 ans: {8,6,7,9} or {5,7,8,9} 35

1.16. You have two arrays A and B of strings. In the array B all element are from A except one. You have find out the string which is extra in B in O(n) time. ex: 36

1.17. Given a m\*n matrix and a person is sitting in (0,0) box, and he has to go to the (m-1,n-1) box of the matrix .And the person can only go to right or down box from its current box position . We need to find out the number of ways he can reach from start to destination box . 36

1.18. Given 2 dimensional sorted array (Both row and column wise sorted) write an efficient code to find median. 36

1.19. You are given an array of integers, say array1 of size n. You have to create another array (say array2) and fill its contents by following this rule: array2[i] will hold the value of the left-most integer from the range array1[i+1] to array1[n-1] such that it is greater than array1[i]. If no such element exists, assign -1. 36

1.20. Given an array with positive and negative numbers find the first continuous subarray that sums to 0. 37

1.21. You have 2 character arrays. The arrays have characters ranging from a-z (all small letters).1. Merge the two arrays, 2. Sort the array. Perform the above in O(N) 38

1.22. Given two sorted arrays. A and B. there are empty spaces in A which is same size of B. Merge them. 38

1.23. Sorted Array Merge Problem: Design an algorithm to merge two sorted positive integer arrays A and B. Size of array A is n and size of array B is m. Array A has m empty blocks(-1) at random places. Merge array B into A such that array A is sorted and does not have empty blocks. 38

1.24. Write two algorithms to remove duplicates from an integer array. 38

1.25. The sorted array is rotated by some factor. And we need to search a key? O(logn) ? 39

1.26. Given a rotated-sorted array find the point of rotation. Complexity less than O(n). 41

1.27. Given a matrix, print it spirally. 42

1.28. There is a matrix where the cost of moving horizontally is 1, vertically is 1 and diagonally is 1.1. Now given two points, what is the number of shortest paths between these two points? 42

1.29. Find third largest element of an array in a single pass? 42

1.30. Given a 4GB file of numbers. You are asked to find the product of K largest numbers. The size of the main memory is 1 GB. Give an efficient method to find. 43

1.31. Given a array find the sub array with maximum sum. 44

1.32. We have two sorted int arrays: a[] --> size N --> contains N elements; b[] --> size 2N --> contains N elements, and N vacant locations. Write an algorithm of complexity O(n) such that b[] contains elements of a[] and b[] in ascending order. 44

1.33. Given two sorted arrays a[] and b[]. Merge them and answer should be stored in a[] with minimum complexity 44

1.34. Given two arrays a[] = {1,3,2,4} b[] = {4,2,3,1}both will have the same numbers but in different combination. We have to sort them. The condition is that you cannot compare only elements within the same array. 45

1.35. Generate the intersection of two sorted sequences 46

1.36. Find two numbers(a and b) from integer array such that a + b = x, where x is input along with array. 47

1.37. Given a set of n integers and an integer x. Design an algorithm to check whether k integers add up to x in the given set. The complexity should be O(n^(k-1) \* logn) 47

1.38. Design an algorithm to find all elements that appear more than n/2 times in the list. Then do it for elements that appear more than n/4 times. 47

1.39. Finding the Median of unsorted array in Linear Time 47

1.40. Given an array of integers for each elemnt in the array find the closest greatest elemnt to the right. Closest means the distance between two elements array indices must be minimum. Problem is to find index of closest greater element to its right. 48

1.41. Given an array A[] and a integer num. Find four no.s in the array whose sum is equal to given num. Brute force :- O(n^4) I solved it in O(n^3) 49

1.42. Given an array which consists of elements in the following form : ->All the adjacent elements differ only by value -1 or +1. ->You are given an element. You need to search for its index. 51

1.43. Given a doubly linked list with just 3 numbers 0,1,2 . Sort it 51

1.44. Dutch National flag problem 52

1.45. How to find the max product of three numbers out of all elements of an array of integers. 52

1.46. You are given some denominations of coins in an array (int denom[])and infinite supply of all of them. Given an amount (int amount), find the minimum number of coins required to get the exact amount. What is the method called? 53

1.47. Find the maximum sum subsequence in an array with positive nos such that no two nos are adjacent. Should explain code, write the code with base conditions properly, and give all possible test cases. 54

1.48. Given an array all of whose elements are positive numbers. Find the maximum sum of a subsequence with the constraint that no two numbers in the sequence should be adjacent in the array. So 3 2 7 10 should return 13 (sum of 3 and 10) or 3 2 5 10 7 should return 15 (sum of 3, 5 and 7) 56

1.49. Find the second largest element in an array with minimum no of comparisons and give the minimum no of comparisons needed on an array of size N to do the same. 57

1.50. Given a string, find the first un-repeated character in it? Give some test cases 57

1.51. Given an array which is sorted in ascending order upto kth element and then in descending order. Find the maximum element. 59

1.52. Sort an array of n positive integers containing n/2 sorted integers in first and second-half? 59

1.53. How to find duplicate element (only one element is repeated) from an array of unsorted positive integers.. 60

1.54. Given a 2d array sorted in increasing order from left to right and top to bottom, what is the best way to search for a target number? 60

1.55. We have a N X N matrix whose rows and columns are in sorted order. How effeciently can we find the median of those N^2 keys ? 62

1.56. Find 2 numbers that add to given sum in an array. 62

1.57. You are given a dictionary of all valid words. You have the following 3 operations permitted on a word: delete a character, insert a character, replace a character. Now given two words - word1 and word2 - find the minimum number of steps required to convert word1 to word2. (one operation counts as 1 step.) 62

1.58. Merge two sorted integer arrays in place where the second one has space for the elements in the first. 63

1.59. Given a set of strings, check whether it is possible to chain all of them. Two strings can be chained iff s1[n] == s2[0] || s2[0] == s1[n] 63

1.60. Given huge file having list of numbers find the largest possible sum of any k numbers 63

1.61. Given an array of strings, we have to write a function called reverseStringSort which have to reverse the given strings, sort it and again reverse it finally before displaying it. 63

1.62. Write C code to implement the strstr() (search for a substring) function. 64

1.63. Given a list of words, identify words which are anagrams of each other, and print them out as sets of anagrams. 66

1.64. Write a function to check if the two strings given are anagrams are not 66

1.65. Median of two sorted arrays 66

1.66. Implement a func node \*(char \*word){} which returns a link list of words that are anagrams with the input word..if no anagrams found return NULL and add that word to the link list 72

1.67. Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1. 72

2. Linked List 74

2.1. Find first two numbers whose sum equals a given number in infinite length (stream of numbers) singly linked list. 74

2.2. Given two lists, each containing numbers, how would you find the intersection of these two lists? What if these two lists are read from a huge file that cannot fit in memory? 75

2.3. Find the intersection point of 2 linked list (without hashing at a lesser complexity preferably O(m+n)). 75

2.4. Reversing of double linked list 78

2.5. Reverse a linked list 78

2.6. Pairwise swap elements of a given linked list 79

2.7. Reverse a Linked List in groups of given size K 80

2.8. Given multiple sorted lists, merge them into a new sorted list. 82

2.9. Input linked list: 83

2.10. Find the kth element from last in link list 83

2.11. You are given a singly linked list. On seeing a node, if it has a node with greater value than itself on its right delete the seen node. Return the head of the result singly linked list. 83

2.12. Sort a doubly link list in less then O(nlogn) time 84

2.13. Random pointer is present in every node of the linked list . And they will be pointing to any of the node of the list . We need to clone this list and return it . 87

2.14. Copy a linked list with next and arbit pointer 87

2.15. Merge 2 sorted linked list into another sorted linked list, then do it in O(N) time and 89

2.16. How do you test the numbers present in a linked list is palindrome or not? (List is a singly linked list) 91

2.17. Convert the matrix into a linked list (singly or doubly). Nodes have two pointers - right and down. 95

2.18. Detect and Remove Loop in a Linked List 95

2.19. How to find out middle element from a looped single linked list 101

2.20. Find the middle element in a singlely linked list 101

2.21. How will you add two nos represented as a singly linked list ( 2->9->1->7+5->1->7= 3->4->3->4 ) 101

3. Trees, Binary Trees, and BST 103

3.1. Deserialize a Binary Tree from it's Pre & In order traversals 103

3.2. Find Height of a tree without using recursion? 105

3.3. Given a binary tree, write a code that returns the difference between sum of nodes at even level and sum of nodes at odd level. Root is considered at level 0. (written test question. coding.) 106

3.4. Given an N-Ary tree, WAP to find the minimum depth of the tree. 107

3.5. Given a Binary Search Tree, write a program to print the kth smallest element without using any static/global variable. 107

3.6. Given a modified BST where each node carries extra information of the total number of nodes below it. Find the Kth smallest number in O(logn) time. 115

3.7. Given a binary tree, convert into a doubly linked list.The list must be as if the tree is traversed in zig-zag order from top to botton. (left to right in one level and right to level in the next) 116

3.8. Given a tree, in addition to the left and right pointer, it has a third pointer, which is set to NULL. Set the third pointer to a node, which will be the successor of the current node, when the tree is traversed in the zig-zag order. In other words, if we traverse the tree using this third pointer alone, then we will be traversing the tree in the zig-zag order. 119

3.9. Given a binary tree, print all root-to-leaf paths 119

3.10. Given a Tree (not binary Tree), print only leaf nodes with it's path from Root 120

3.11. Given a binary tree which contains values at each node, find whether the path exist from root to the "LEAF NODE" such that sum of the values of the path nodes is equal to the GIVEN SUM. If so return true or else return false 120

3.12. Given a binary tree, find the path from the root to all nodes whose id is multiples of five. 120

3.13. Given a n-ary tree. A random leaf node will be selected. Imagine that you are now holding the tree with your hand from that node. All other nodes will now fall under gravity. Write a function to perform this transformation. 120

3.14. Given a tree, parse the tree using breadth first search and then find the last element in the tree. After finding the last element, replace that element with the root node. Eg. Input: 121

3.15. You are given a tree where child nodes point to parent (there is no link from parent to child). Give any two nodes in the tree, write a function to find the lowest common ancestor. Node\* LCA(Node \*p, Node \*q){ …} 121

3.16. Given two nodes of tree .Find their first common ancestor 122

3.17. Given a BST and two values m and n . We need to find out all the nodes whose values are in range of m and n . 125

3.18. Remove BST keys outside the given range 125

3.19. Given a binary tree, write a code to update each node with the sum of values of its subtree. 128

3.20. Given a Binary Tree ( Comprising of +ve & -ve numbers ), represent each node of the tree with sum of its LeftSubTree & RightSubTree 129

3.21. Given a binary search tree and a value X, find the node with value immediately greater than X. 129

3.22. Find the maximum subtree in a given binary tree which is a binary search tree 130

3.23. If a binary tree is a binary search tree. 132

3.24. Write the code to test if a given binary tree is balanced or not? Write/give test cases 134

3.25. To get mirror image of a binary tree. 134

3.26. Convert Sorted Doubly Linked List into a Binary Tree 135

3.27. WAP to check a tree is symmetric. 136

3.28. Find median of BST. 136

3.29. Given an n-ary tree of resources arranged hierarchically. A process needs to lock a resource node in order to use it. But a node cannot be locked if any of its descendant or ancestor is locked. You are supposed to: -> write the structure of node -> write codes for 137

3.30. Find diameter of a binary tree in O(n) 138

3.31. Given a binary tree. Find the minimum element at a given depth. 139

3.32. Vertical Sum of a Binary Tree OR Vertical split of a binary tree 140

3.33. Given a Node in a binary tree . We need to find out all the nodes at K distance from that node . 141

3.34. For the given number need to find out the possible BST. eg, if the given number is n means we should find BSTs using 1,2..n. n=5 means figure it out using 1,2,3,4,5, how many BST we can make? 141

3.35. There is a Directory in which there are subdirectories and recursively have subdirectories. We need to find out is any subdirectory is pointing to its any of its ancestor. 141

3.36. Given a no. find all pairs of elements in a balanced bst, which sum to this number. Make sure you don’t return same pair again. 141

3.37. construct double linklist out of a BST 141

3.38. Print the nodes of binary tree in on sorted order 141

3.39. Two BST were given and find the largest common bst that exists between these two 141

3.40. Print a tree layer by layer 141

4. Algorithms 141

4.1. Types 141

4.2. you have an array of integers, find the longest subarray which consists of numbers that can be arranged in a sequence, e.g.: a = {4,5,1,5,7,4,3,6,3,1,9} max subarray = {5,7,4,3,6} 142

4.3. You have given a file containing sentences. Now you are given a sequence of characters. You have to find the starting location of each word containing one of the permutations of the word. 142

4.4. We have a job sequence like this. 142

4.5. Edit Distance ? 142

4.6. In a plane we are given latitude, longitude coordinate, and we are also given a point (having lat, long value). We need to find out the nearest point, in most efficient way. 142

4.7. Given a large file containing 10 bit integers, and a limited memory, sort the entire file and store back to file. Constraints: You cannot read entire file in main memory at a time as memory is limited (small) 143

4.8. How will you find n most frequently occurring patterns in a text file. What data structures would you use? Here, a pattern is not a single word but rather a sequence of words. For instance, "this is a" could be a frequently occurring pattern in the file. 143

4.9. you have a Log file that stores the "users id", "time"(irrelevant), and "webpage visited." Find the most common three-sequence webpage visited by the users. For example, you have a log file 143

4.10. Jump Game: Given an array start from the first element and reach the last by jumping. The jump length can be at most the value at the current position in the array. Optimum result is when u reach the goal in minimum number of jumps. 144

4.11. Given millions of time intervals, compute the maximum weights and the time when maximum weights occurred. Time interval is defined as followings (s: starting time, e: ending time, w: weight). And pls note that s, e, and w are not necessary integer. 146

4.12. There is a 3\*3 grid with 8 numbered tiles and 1 vacant space.Starting with initial grid arrangement, devise a strategy to reach final grid arrangement. 146

5. Math 146

5.1. Output all dates "Fridays, 13th" in the format dd.mm.yyyy starting from 1st Jan 1900 (Monday) 146

5.2. There are three arrays of numbers A,B and C. You have to find out all tuples <a,b,c> such that a-b = c where a is from A,b is from B and c is from C. 148

5.3. Convert a double-precision number to rational, i.e.: 0.125 -> 1/8 don't care about arithmetic overflow 149

5.4. A stream of 1's and 0's are comming .At any time we have to tell that the resultant number from the binary digits till that point is divisible by 3 or not .For eg: let's see one example.Let 1 come (not div by 3) .then 1 come so resultant binary number is 11(3) which is divisible by 3 , then 0 come make it to 110(3) which is divisible by 3, then 0 come make it to 1100(12) which also divisible by 3 . 150

5.5. Given n lines in the plane (for simplicity assume no 3 lines intersect at one point). Count the total number of triangles in the plane created by these lines. Observe that smaller triangles may be part of larger ones. Look here for example: http://farm8.staticflickr.com/7021/6465828833\_15e7447992\_z.jpg 150

5.6. Design an algorithm to find the immediate greater number to a given number, such that the result has the same digits as the given input number. 151

5.7. Write a function that takes an integer and returns a char array that contains the -2 base representation of the given integer. 152

5.8. Write a function to check if an integer is square or not? Ex, 49 is square, 48 is not. 153

5.9. given an array , find three numbers a,b and c such that a^2+b^2=c^2 .. I have given algo of o(n^2) complexity. But he is expecting still better algo. 153

5.10. Given a number N find the next number M which is greater than N and the difference between M and N is small and M consists of same digits as N ! i.e M is an anagram of N ... 154

5.11. Given a number, convert it to minimal sum of squares. N can be represent as “x^2 + y^2 + z^2” or “a^2 + b^2”. But here the exception is minimal i.e. the answer will be “a^2 + b^2” 154

5.12. Given a big unsorted list of 64-bit integers, find an element not in list 154

5.13. Itoa 154

5.14. How do you convert a decimal number to its hexa-decimal equivalent.Give a C code to do the same 155

5.15. Print if a given integer is a palindrome or not 156

5.16. How one would implement int Power? 156

5.17. Given a rectangle of dimensions b\*h, and a input no N, find the size(side length) a such that N squares of size a can fit into the rectange. Optimize based on the minimum wasted of space in the rectangle b\*h 156

5.18. Given a number, convert it to minimal sum of squares. 156

5.19. Given 3 prime numbers and an integer k, find the kth number if all the nos which are having these 3 prime numbers as their factors are arranged in increasing order. 157

6. Data Structures 158

6.1. Give an algorithm for finding duplicate parenthesis in a expression. (( a + b ) \* (( c + d ))) 158

6.2. Code up a system that will accept a series of telephone keypresses and return a list of possible names from a supporting data structure. Describe both the data structure and the insert and search methods 158

6.3. Write efficient code for Singleton class. I wrote the double-checked locking version. 158

6.4. Write a C program to create a bitmap of any size as determined by user. Say user says 64k bitmap, and then create 64k long bitmap. Have set and unset methods. 159

6.5. Simple Reference counted smart pointer 159

6.6. Find the element with the middle value among all elements of the stack...you can you extra space but complexity should be minimum. Problem is to find the median of all the elements in the stack. 160

6.7. Code/Define Algorithm to find if a given string has balanced parentheses, where we have 3 types 1. ( ) 2. [ ] 3. { }. 161

6.8. Write code to retrieve max-valued element from a stack. Time complexity must be O(1). 161

6.9. Given an infix expression convert into postfix 161

6.10. Write code to retrieve max-valued element from a stack. Time complexity must be O(1). 162

7. Design 163

7.1. I have four operations, Set(i) sets the ith bit, Unset(j) unsets the jth bit, Set(i,j) sets the bits b/w i and j, Unset(i,j) unsets the bits b/w i and j. Design a data structure for this problem with minimum time and space complexity. 163

7.2. Design the system for threater reservation system. for example Seat or chairs are organized in the form of rows and columns. When the first person come and book the ticket need to provide a seat on the middle of the last row. When next person come we have to provide empty space between the existing audience and book the ticket for the set of people. How we will design this system? 163

7.3. Given the classes below how would you improve the class design and their relationships 163

7.4. Given a Parking lot having fixed number of slots, where a car can enter the lot if there is a free slot and then it will be given the direction of the free slot. When exits the car has to pay the fees for the duration of the time the car is in the slot. How you will design the data model and functional model. 163

7.5. Design classes to calculate the price of a pizza with different crusts, toppings and sauces 164

7.6. Design and create a Collections library. Make it flexible from two aspects 164

7.7. Assume you have a monolithic web site and you are asked to rearchitect the website 164

7.8. OO Design for a restaurant reservation system. 164

7.9. Design a DFS that supports mount, open, read, write, close and unmount 164

7.10. How would you architect an online website? 164

7.11. Suppose you are writing a crawler meant to crawl the entire web. How would you avoid reindexing the same page. 164

7.12. Given a database table of order entries and n service instances (S0..Sn-1) processing these entries. Instance Si processes a record if the order-id%n = i. Do you see any issues with this design? 164

7.13. Given a series of stock prices in an array. Pick when to buy and when to sell to gain max profit. 164

7.14. Discussion on various data structure where the following operations could have minimal cost: 164

7.15. The best data structure to store the phone book. Or dictionary 164

7.16. Design an API for this scenario. There is a onetime input of a set of data. The user works on an interface that has two options- a) Find(x) which returns x if found. b) Findkthmin() This interface is used several hundred times a day. Describe the data structure& algorithm that you will use. 164

7.17. A "Most efficient data structure" is designed to optimize the following operations. Pop, Push, Min. The best possible time-complexities with no extra space, respectively would be? 165

7.18. You need to implement two functions of hashing int get(int key) // will return value at key void put (int key,int value) // put the vaue at index key, both functions should be of order O(1) and there is an upper limit n on size, such that at the max only n elements will get stored and if there is a conflict then you need replace the least recently used element 165

7.19. Design an algorithm to perform operation on an array Add(i,y)-> add value y to i position sum(i) -> sum of first i numbers we can use additional array O(n) and worst case performance should be O(log n) for both operation 165

7.20. Design a data structure for a phone address book with 3 fields name, phone number , address. 166

7.21. Design pizza topping 166

8. Puzzles 166

8.1. There are 100 prisoners, and a officer of them. Now the officer gave the command to the prisoner that next day they will be going to wear a hat which they will not be know its color. But its color will be either Red or Blue. And he says that the entire prisoner will be standing in a line. And then the officer will start asking the color of the prisoner one by one from the back. Whichever prisoner says the wrong color of his hat, gets shoot .So now we have to find out what strategy should the prisoners should apply to safe maximum prisoners. 166

8.2. Given a set of cubes with sides colored with random colors. Find if the cubes can be stacked one on top of other such that each of the four lateral sides has the same vertical color. 167

8.3. Given n red balls and m blue balls and some containers, how would you distribute those balls among the containers such that the probability of picking a red ball is maximized, assuming that the user randomly chooses a container and then randomly picks a ball from that. 168

8.4. Given a cube of size n\*n\*n (i.e made up of n^3 smaller cubes), find the number of smaller cubes on the surface. Extend this to k-dimension. 168

8.5. 100 rooms in a prison. All doors are closed initially. A policeman on round i will toggle the doors that are in multiples of i. After 100 rounds, what doors will be open? 168

8.6. Given a clock, take time as input in any format; provide a formula to calculate the angle between the minute and hour hand. 168

8.7. There are 25 horses among which you need to find out the fastest 3 horses. You can conduct race among at most 5 to find out their relative speed. At no point you can find out the actual speed of the horse in a race. Find out how many races are required to get the top 3 horses. 168

8.8. There are n petrol bunks arranged in circle. Each bunk is separated from the rest by a certain distance. You choose some mode of travel which needs 1litre of petrol to cover 1km distance. You can't infinitely draw any amount of petrol from each bunk as each bunk has some limited petrol only. But you know that the sum of litres of petrol in all the bunks is equal to the distance to be covered. ie let P1, P2, ... Pn be n bunks arranged circularly. d1 is distance between p1 and p2, d2 is distance between p2 and p3. dn is distance between pn and p1.Now find out the bunk from where the travel can be started such that your mode of travel never runs out of fuel. 169

9. Miscellaneous 169

9.1. Write a function to reverse a UTF-8 encoded string in-place. 169

9.2. There is a document and some key words are given. Find the minimum length of substring in the document which contains all the keywords. The substring can contain key word in any order. 169

9.3. There is a straight road with 'n' number of milestones. You are given an array with the distance between all the pairs of milestones in some random order. Find the position of milestones. 169

9.4. Given a large list of 2D Points (x,y) find the k closest points to a target P(x,y). 170

9.5. Given 0\*1\*, find index of first 1. Ie Given an infinite size array with only 0s and 1s and sorted. find the transition point where 0s end or 1s start 170

9.6. Write a function which takes as parameters one regular expression(only ? and \* are the special characters) and a string and returns whether the string matched the regular expression. 170

9.7. Given int n, the total number of players and their skill-point. Distribute the players on 2 evenly balanced teams. 170

10. Theory 171

10.1. Under which conditions the default assignment operator is not generated by the compiler for your class ? 171

10.2. What is the time and space complexities of merge sort and when is it preferred over quick sort? 171

10.3. What is Map and what are the different ways of implementing map. Give pros and cons of each approach 171

10.4. Difference btw process and thread? 171

10.5. What is a tree data structure 172

10.6. difference between tree and graph 172

10.7. We have a text file, how to choose between a hash-based index and a tree-based index? 172

10.8. Fork vfork exec 172

10.8.1. Fork 172

10.8.2. Vfork 173

10.8.3. Exec 173

10.9. IPC Linux 173

10.9.1. Pipes 173

10.9.2. Shared Memory Linux 173

10.9.3. Memory Map 174

10.10. Pthread 176

11. Permutation and Combinations 176

11.1. give a \*recursive\* algorithm to print all permutations of 'n' consecutive integers in lexicographic order, i.e.: 176

11.2. Given an integer array. e.g. 1,2,3,4,5. Compute array containing elements 120,60,40,30,24 (2\*3\*4\*5,1\*3\*4\*5, 1\*2\*4\*5, 1\*2\*3\*5, 1\*2\*3\*4) 176

11.3. Find all the permutation of the given string? But take care of duplicate characters. 177

11.4. given a set of letters and a length N, produce all possible output.(Not permutation). For example, give the letter (p,o) and length of 3, produce the following output(in any order you want, not just my example order) 177

11.5. given a number, come up with all of the possible ways to insert '+' and '-' in that number. for example given 123, possible answer would be 177

11.6. Given a string "abcd" print combinations of length n. Example if n= 3 print abc, abd, acd, bcd. Was asked to use recursion. 178

12. Yahoo 1-8 questions 178

12.1. Given an 4n X 4n Matrix, where n is a positive integer taken as input. Imagine the matrix consisting of two interleaved coils whose centers are at the centre of the matrix. Implement a java program which takes an integer (n) as input and prints the two coils in two seperate lines. 178

12.2. Suppose there is an array with numbers : 1, 14, 5, k, 4, 2, 54, k, 87, 98, 3, 1, 32. Output for this can be assuming k =20 1,14,5,4,2,3,1,k,k,54,87,98,32. Now sort this array in a way all k are in middle and all values on left of k are smaller (in any orer) and on right are larger (in any order) Note: k is an integer value within range of 1 – 32768 180

12.3. If there is a million data in file(assume integers). The memory is enough to hold all the data. After loading all the data into data structure , we need to insert 500 new integers after the 10000th element. What Data structure to use and how to use? 181

12.4. How would you design an Excel sheet's Data structure. You need to perform operations like addition. The excel sheet is very sparse and is used to store numbers in the range 1-65K. Index for a cell is known. 181

12.5. How to sort a 1000 GB file with ram size is 4 GB only. Which algorithm or data structure we need to use to sort these files? 182

12.6. File-1 is having 5 million strings and File-2 is having 1 million strings. Give an Algo to remove duplicates and merge these files (Need not be sorted) into File-3. 182

12.7. given m x n matrix print all the possible paths top to down. 182

12.8. dbx and how to debug a multi threaded application 184

12.9. I have an arrayList A which contains say 2,3,5,7,8. I have another arrayList B which contains 1, 3 Now taking the elements of B as the locations, I need to remove the elements of A present in that locations. So, basically I need to remove the element 2(position 1) and 5(position 3) from A. How to achieve it as we know that once one element got removed from an arrayList,the positions will be auto adjusted. 184

12.10. You are given N points in a X-Y plane. Find the two closest points out of them. 184

12.11. Given a few numbers, how will you push them into a stack such that whenever I pop the top most element from that stack, I get the minimum number from the bunch. Also he wanted me to tell the pop push algorithm such that the order of insertion and popping should be O(1). 188

12.12. Given a set of positive integers S = { a1,a2,a3,...,an} find two subsets s1 and s2 of A such that S2 = S - S1 and difference of | sum(S1) - sum(S2) | is minimum. 189

12.13. You are blindfolded and placed in front a table with two jars. One jar has 50 red balls and other has 50 blue balls. What should be your strategy so that you pick up the red ball with more than 50% probability. 189

12.14. In 1000 wine bottles stack 10 are poisoned given 10 rats what is the minimum number of tries to find the poisoned one. Rat dies once it licks the poisoned wine. 189

12.15. Celebrity problem: You have a room with n people. A celebrity walks in. Everyone knows the celebrity, the celebrity knows no one. Non-celebrities may/may not know anyone in the room. Give an algorithm to find the celebrity. Discuss the complexity. 189

12.16. This is a very simple question. But im not sure why the answer is the way it is. 189

12.17. Difference between == and === in JScript, Write a method to detect mouse hovering over a text box using CSS style sheets 190

12.18. Design a simple Address book Web application, Discuss all aspects of creating the App , starting from UI to Back-end databases 190

12.19. Design a client-server application in the Application layer of TCP/IP 191

12.20. Design mine-sweeper. 191

12.21. What is Hashing. 191

12.22. There is external file with 5billion numbers. how will u sort. Wht if 1billion pairs? 191

12.23. Have 5million key-value pairs wht datastruct u will use? 191

12.24. What does public static void main means ..how it is called? 191

12.25. Wht is oop and comparision with traditional method. 191

12.26. Wht is abstraction? 191

12.27. Diff bet abstract class and interface and when u use one?" 191

12.28. A forgetful professor wants to know the names of all the students in a class of strength 'n'. For this, he makes 'm' students stand up at a time. He remembers the names of all the students who are standing and the ones who are sitting but doesn't remember the name of each student individually. He does this operation 'k' number of times and tries to infer all the names. What can be the value for 'k' when n=13 and m=5 191

12.29. Given an arry Arr[N] of integers and a function int func(int x) that takes an integer and returns either 0 or 1 (depending on some property of the integer). Give the most efficient algorithm to store the numbers in the array in such a way that all numbers that return 1 should come before all numbers that return 0, when called upon by the function func 191

12.30. Design an algorithm to sort an array whose first n-sqrt(n) elements are already sorted. What is the complexity of the algorithm. 192

12.31. you have to do the queue operation when you have a single stack..for example:-suppose there is a telephone bill populating in a stack now you have to process each request in a fifo manner.you can perform only push pop operation and allocating extra memory is not allowed. 192

12.32. find missing numbers in given billion number.( numbers lie between 1-k) 193

12.33. Sort 10 GB file using 2 GB memory. and complexity 193

12.34. Given 2 Tree, find out if they r exactly same or not 193

12.35. Given a graph of price variation of a stock over a period of 12 months, return the ideal time to buy and sell, for maximization of profit. Time duration is not constraint. Time duration needn't be minimum it can even be maximum 12 motnhs 193

12.36. Given a sequence of charecters , print consequtive sequence of charecters alone in a line, other wise print it in a new line. 193

12.37. How to find the depth of a tree. Time and Memory optimization is emphasized 194

12.38. What are the various ways to swap 2 variables 194

12.39. Given a file, return the Top 5 frequently occuring list of words 195

12.40. Design a Chess Game 195

12.41. Design Snake and Ladder Game 195

12.42. Given a white background with a black random shape scattering around. Describe and code an algorithm to count the number of black shapes. You can assume input to be an array of 3xMxN in RGB color space. 195

12.43. Given one unsroted integer array, find out all the unique element in the array. 195

12.44. Given an integer n , you have to print all the ways in which n can be represented as sum of positive integers 195

12.45. write a program to display the no. of 1,2,3,4... lettered words in paragraph. 195

12.46. Count the number of unique element in array of numbers in minimum time complexity. 196

12.47. Given an array of length N. How will you find the minimum length contiguous subarray whose sum is S and whose product is P . HereS and P will be given to you. 196

12.48. Implement a search engine. The input will be 100 text files. Return the paragraph and the file in which the keyword appears. 196

12.49. Implement a phone directory 196

12.50. Implement Twitter's trending topic. The input is a text file. Implement it in C. 196

12.51. Given a stack S, write a C program to sort the stack (in ascending order). You are not allowed to make any assumptions about how the stack is implemented; the only functions allowed to be used are: Push, Pop, Top, IsEmpty, IsFull. 196

12.52. There is a sequence of increasing numbers that have the same number of binary 1s in them. Given n, the number of 1 bits set in each number, write an algorithm or C program to find the n’th number in the series. 196

12.53. You are given have a datatype, say X in C. Determine the size of the datatype, without declaring a variable or a pointer variable of that type, and, of course without using the sizeof operator. 197

12.54. Write a C Program to reverse a stack in place using recursion. You can only use the following ADT functions on stack: IsEmpty, IsFull, Push, Pop, Top. you can not use extra stack or any other data structure 197

12.55. Write a program to find and print the 1500’th ugly number. 198

12.56. Write a method to generate a random number between 1 and 7, given a method that generates a random number between 1 and 5 (i.e., implement rand7() using rand5()). 202

12.57. Given two sets, write a function to provide the union of them. 203

12.58. You are given a list of Ball objects. Each Ball is either Red or Blue. Write a function that partitions these balls so that all of the balls of each color are contiguous. Return the index of the first ball of the second color (your result can be Red balls, then Blue balls, or the other way around). In haskell, you’ll probably want to return a ([Ball],Int)." 203

12.59. Given a linked list, split it into two lists - one for the front half and, one for the last half. If odd number of elements are present the extra node should go to the front list 203

12.60. Remove duplicates from a linked list 203

12.61. You are given: 3 types of vehicles: Motorbike, Car, and a special type of car for the handicapped. 3 types of parking: Motorbike parking, Car parking, handicapped car parking. Motorbikes and cars can only park in their designated parkings, while the handicapped cars can park either in their own parking or the regular car parking. How would you model this as classes? Explain your methods. 204

12.62. Find the missing element in a sorted array in most optimum running time (O(log n)) 205

12.63. Given a balanced BST tree, write a function to replace the root with a node that belongs to the original tree. 205

12.64. Given n non overlapping intervals and an element. Write a program to find the interval into which this element falls. 205

12.65. Design a stack which supports push, pop, min and max operations in O(1). 205

12.66. There is a blank disc. You are given two colors of paint (black and white) . A sensor can recognize the color painted on the disc and produce an output. Paint the disc in a way such that you can find the direction of rotation by looking at the output (BWBWBWBW... etc). Find the minimum number of sectors you will need to paint" 205

12.67. An array of size n, has n/2 unique elements and n/2 occurences of an element. Find the non-unique element in linear time? 205

12.68. You are provided with a stream of numbers, design a data structure to store the numbers in the stream along with their no. of occurrences. 206

12.69. Count number of lines, characters and words in a file (Given that we don’t have much access to flashy java methods like readline, String methods like indexOf etc.) 206

12.70. Write a function which takes as parameters one regular expression (only ? and \* are the special characters) and a string and returns whether the string matched the regular expression. 206

12.71. "You are given some denominations of coins in an array (int denom[])and infinite supply of all of them. Given an amount (int amount), find the minimum number of coins required to get the exact amount. What is the method called?" 207

12.72. "Given a cube of size n\*n\*n (i.e made up of n^3 smaller cubes), find the number of smaller cubes on the surface. Extend this to k-dimension" 207

12.73. "There are four dogs, each at the counter of a large square. Each of the dogs begins chasing the dog clockwise from it. All of the dogs run at the same speed. All continously adjust their direction so that they are always heading straight towards their clockwise neighbor. How long does it take for the dogs to catch each other? Where does this happen? (Hint: Dog’s are moving in a symmetrical fashion, not along theedges of the square)." 207

12.74. Given two lists write a function which returns a list which is the intersection of the two lists 207

12.75. "Three strings say A,B,C are given to you. Check weather 3rd string is interleaved from string A and B. 207

12.76. "Given that you can take one step or two steps forward from a given step. So find the total number of ways of reaching Nth step." 208

12.77. Count the number of set bits in a number. Now optimize for speed. Now optimize for size 208

12.78. An array A[1....n] is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is A[i] > A[j]?". However you can answer questions of the form: "is A[i] = A[j]?" in constant time. 208

12.79. You are given with three sorted arrays (in ascending order), you are required to find a triplet (one element from each array) such that difference between their positions is minimum" 208

12.80. Calculate the Resistance in ohm's that a knights move would require on an infinite plane of resistors of unit resistance. 208

12.81. Given a set of coin denominators, find the minimum number of coins to give a certain amount of change. 208

12.82. Get the width of a binary tree 208

12.83. Suppose you have an NxN matrix of positive and negative integers. Write some code that finds the sub-matrix with the maximum sum of its elements. 208

12.84. Given a binary tree with the following constraints: a) A node has either both a left and right child OR no children b) The right child of a node is either a leaf or NULL write code to invert this tree. 208

12.85. Given the sequence S1 = {a,b,c,d,...,x,y,z,aa,ab,ac.... } and given that this sequence corresponds (term for term) to the sequence S2 = {1,2,3,4,....} Write code to convert an element of S1 to the corresponding element of S2. Write code to convert an element of S2 to the corresponding element of S1." 208

12.86. "Given N computers networked together, with each computer storing N integers, describe a procedure for finding the median of all of the numbers. Assume that a computer can only hold O(N) integers (i.e. no computer can store all N^2 integers). Also assume that there exists a computer on the network without integers, that we can use to interface with the computers storing the integers." 208

12.87. "Write a function that returns a node in a tree given two parameters: pointer to the root node and the inorder traversal number of the node we want to return. The only information stored in the tree is the number of children for each node" 209

12.88. Write a function to find the next prime number after a given number. 209

12.89. "A man has two paper cubes on his desk. Every day he arranges both cubes so that the front faces show the current day of the month. What numbers are required on the faces of the cubes to allow this for all possible days in the calendar?" 209

12.90. A band is going in the street with a constant speed. Someone in the last row has a dog. The dog runs ahead, reaches the front row of the band and gets back to it's owner. The dog's speed was constant all the way and while it was running the band passed 50 feet. Find the length of the dog's path,if the distance between the front and the rear row of the band is 50 feet. 209

12.91. There are 100 doors in a row that are all initially closed. You make 100 passes by the doors starting with the first door every time. The first time through you visit every door and toggle the door (if the door is closed, you open it, if its open, you close it). The second time you only visit every 2nd door (door #2, #4, #6). the third time, every 3rd door (door #3, #6, #9), etc, until you only visit the 100th door. What is the state of each door after the last pass? 209

12.92. An apple is in the shape of a ball of radius 31 mm. A worm gets into the apple and digs a tunnel of total length 61 mm, and then leaves the apple. (The tunnel need not be a straight line.) Prove that one can cut the apple with a straight slice through the center so that one of the two halves is not rotten. 209

12.93. Given two cubes, how will you represent all the dates in a month by using numbers 0 to 9? You can exchange the cubes and rotate them as you wish!!! 209

12.94. You have a machine which can do only multiply by 2, divide by 2 and Addition of 2 numbers. Write a detailed algorithm to multiply any two numbers, in this kind of a machine. 209

12.95. Write c++ working code for Given a large number with many digits, propose a method or data structure to efficiently store them. Addition, subtraction, mult, division should be supported by your design. 209

12.96. Let f(k) = y where k is the y-th number in the increasing sequence of non-negative integers with the same number of ones in its binary representation as y, e.g. f(0) = 1, f(1) = 1, f(2) = 2, f(3) = 1, f(4) = 3, f(5) = 2, f(6) = 3 and so on. Given k >= 0, compute f(k)." 210

12.97. Compute the discrete log of an unsigned integer. 210

12.98. Given a singly linked list, print out its contents in reverse order. Can you do it without using any extra space? 210

12.99. Given a linked list with the following property node2 is left child of node1, if node2 < node1 else, it is the right child. 210

12.100. Given a list of numbers ( fixed list) Now given any other list, how can you efficiently find out if there is any element in the second list that is an element of the first list (fixed list). 210

12.101. An array of size k contains integers between 1 and n. You are given an additional scratch array of size n. Compress the original array by removing duplicates in it. What if k << n?" 210

12.102. Suppose you are getting an infinite binary stream of characters then after any point of time you need to print whether the no is divisible by 3 or not, how will you do that? 210

12.103. An array is of size N with integers between 0 and 1024(repetitions allowed). Another array of integers is of size M with no constraints on the numbers. Find which elements of first array are present in the second array. (If you are using extra memory, think of minimizing that still, using bitwise operators) 210

12.104. How many matches will be played in a knockout tournament between 9 teams get the general formula for n teams? 211

12.105. design a datastructure to represent the movement of a knight on a chess board 211

12.106. Write an algorithm to traverse a knight covering all the squares on a chessboard starting at a particular point. 211

12.107. "There is a temple, whose premises have a garden and a pond. It has 4 idols, each of Ram, Shiv, Vishnu and Durga. The priest plucks x flowers from the garden and places them in the pond. The number of flowers doubles up, and he picks y flowers out of them and goes to offer it to Lord Ram. By the time he reaches to the pond, he finds the remaining flowers also have doubled up in the meantime, so he again picks up y from the pond and goes to Lord Shiv.This process is repeated till all the Gods have y flowers offered to them, such that in the end no flower is left in the pond. Find x and y." 211

12.108. "On a empty chessboard, a horse starts from a point( say location x,y) and it starts moving randomly, but once it moves out of board, it cant come inside. So what is the total probability that it stays within the board after N steps." 211

12.109. Given that you have one string of length N and M small strings of length L . How do you efficiently find the occurrence of each small string in the larger one ? 211

12.110. You are given with three sorted arrays ( in ascending order), you are required to find a triplet ( one element from each array) such that distance is minimum. Distance is defined like this : If a[i], b[j] and c[k] are three elements then distance=max(abs(a[i]-b[j]),abs(a[i]-c[k]),abs(b[j]-c[k]))"Please give a solution in O(n) time complexity 211

12.111. "Given a Data Structure having first n integers and next n chars. A = i1 i2 i3 ... iN c1 c2 c3 ... cN. Write an in-place algorithm to rearrange the elements of the array ass A = i1 c1 i2 c2 ... in cn" 211

12.112. "There is a linked list of numbers of length N. N is very large and you don’t know N. You have to write a function that will return k random numbers from the list. Numbers should be completely random." 211

12.113. "There are a set of 'n' integers. Describe an algorithm to find for each of all its subsets of n-1 integers the product of its integers. For example, let consider (6, 3, 1, 2). We need to find these products : 6 \* 3 \* 1 = 18 6 \* 3 \* 2 = 36 3 \* 1 \* 2 = 6 6 \* 1 \* 2 = 12" 212

12.114. How would you determine if someone has won a game of tic-tac-toe on a board of any size? 212

12.115. "Given two sequences of items, find the items whose absolute number increases or decreases the most when comparing one sequence with the other by reading the sequence only once." 212

12.116. "How many different binary trees and binary search trees can be made from three nodes that contain the key values 1, 2 & 3?" 212

12.117. "Given an expression tree with no parentheses in it, write the program to give equivalent infix expression with parenthesesinserted where necessary" 212

12.118. "Given ships travel between points A and B, one every hour leaving from both ends (simultaneously), how many ships are required (minimum), if the journey takes 1hr 40 mts. How many ships does each ship encounter in its journey, and at what times? 212

12.119. Count the number of set bits in a number without using a loop. 213

12.120. "How would you reverse the bits of a number with log N arithmetic operations, where N is the number of bits in the integer (eg 32,64..)" 213

12.121. Delete a node from a binary tree and balance it. Write code for the former and explain the latter. 213

12.122. "Given a maze with cheese at one place and a mouse at some entrance, write a program to direct the mouse to cheese correctly. (Assume there is a path). Following primitives are given: moveforward, turnright, turnleft, iswall?,ischeese?, eatcheese." 213

12.123. "A car has speed of 72 64 56 in downhill, plain and uphill respectively . A guy travels in the car from Pt. A to pt. B in 4 Hrs and pt. B to pt. A in 4 Hrs and 40 min. what is the distance between A and B?" 213

12.124. "Write a program to print the elements of a very long linked list in ascending order. There may be duplicates in the list. You cannot modify the list or create another one. Memory is tight, speed is not a problem." 213

12.125. "A real life problem - A square picture is cut into 16 squares and they are shuffled. Write a program to rearrange the 16 squares to get the original big square" 213

12.126. Given an array in which elements are unsorted. Write an algorithm that gives two indices n1,n2 such that if you sort just the elements of the array from n1 to n2, then the whole array will be sorted." 213

12.127. given a word,convert it into a pallindrome with minimum addition of letters to it.letters can be added anywhere in the word.for eg if yahoo is given result shud be yahoohay.give a optimize soln 213

12.128. What is Spring IOC? 213

12.129. What is a deadlock and what are some of the ways to avoid a deadlock? 213

12.130. Explain hashcode() and equals() methods? When would you override these methods? What does the hashcode() method in the Object class do? 213

12.131. What is an immutable object? How do you implement one in Java? How to construct an immutable object that has an array/List as one of the instance properties? 214

12.132. "Find the maximum subsequence sum of an array of integers which contains both positive and negative numbers and return the starting and ending indices within the array. 214

12.133. How can we traverse through all the files in a folder and the subfolders. What system calls should be used(in C). 214

12.134. What is the difference between functors, call back functions and function pointers? 214

12.135. Write a function to add an array of numbers. 214

12.136. What's the difference between assignment operator and copy constructor 214

12.137. What's the difference between pointer and reference 214

12.138. What are call back functions? 214

12.139. What is a functor? 214

12.140. Write a unix program to count the number of lines in a text file 214

12.141. What is pi? 214

12.142. What is polymorphism 214

12.143. What is the importance of the keyword static in java 214

12.144. "Write a program to replace string 'us' with 'them' from the following String . Do not replace 'Us' as well as any string containing us $\_="Us? It usually rains when bus comes to us";" 215

12.145. What is reflection 215

12.146. Given an integer, print the closest number to it that is a palindrome - eg, the number "1224" would return "1221". 215

12.147. How to develop a sorted lexicographic tree. 215

12.148. How to find distance between two lines in a 3D plane 215

12.149. Put eight chess queens on an 8×8 chessboard such that none of them is able to capture any other using the standard chess queen's moves 215

12.150. "What data structure would you use to store distances between all the planets in a galaxy. (So there could be like a billion planets) Also steps in connecting a thin mobile client to connect to the server and get distances between one given planet and all the other planets in that galaxy." 215

12.151. "There are given n men and n women. Each woman ranks all men in order of her preference (her first choice, her second choice, and so on). Similarly, each man sorts all women according to his preference. The goal is to arrange n marriages in such a way that if a man m prefers some woman w more than his wife, and w prefers m more then her husband a new marriage occurs between w and m. If w prefers her husband more, then she stays married to him. This problem always has a solution and your task is to find one." 215

12.152. "Johnny was asked by his math teacher to compute nn (n to the power of n, where n is an integer), and has to read his answer out loud. This is a bit of a tiring task, since the result is probably an extremely large number, and would certainly keep Johnny occupied for a while if he were to do it honestly. But Johnny knows that the teacher will certainly get bored when listening to his answer, and will sleep through most of it! So, Johnny feels he will get away with reading only the first k digits of the result before the teacher falls asleep, and then the last k digits when the teacher wakes up. Write a program to help Johnny to compute the digits he will need to read out." 215

12.153. "The Chef has one long loaf of bread. Let us say, of length 1. He wants to cut it into as many little loaves as he can. But he wants to adhere to the following rule: At any moment, the length of the longest loaf which he possesses may not be larger than the length of shortest one, times some constant factor. Every time, he is only allowed to cut exactly one loaf into two shorter ones. 215

12.154. Your program will take as an input 'n' coordinates of type {(X1,Y1,Z1), (X2,Y2,Z2), (X3,Y3,Z3),...(Xn,Yn,Zn)} and from these 'n' coordinates print a list of 's' coordinates (where 's' is another input parameter less than 'n') which are closest to the origin (0,0,0) and a list of 't' coordinates (where 't' is another input parameter less than 'n') points closest to each other. Your solution should use an optimal strategy and minimal time / space complexity 216

12.155. "int \* p= NULL; p = (int\*) malloc(0); what will be the value of p?" 216

12.156. "if i write main like the following ways. Which one will compile and way? 216

12.157. "to print 216

12.158. How will u find the min element in the stack with O(1) . Space constraint also has to be considered while designing it . 217

12.159. Give an string, return the first non-repetitive character. 217

12.160. "Consider an array of positive and negative integers. We want to find a slice of this array (i.e. a sub-array of consecutive elements) with at least two elements, such that the sum of the elements in this slice is equal to 0. The size of the slice can be anything (i.e. from 2 up to the length of the original array), and we don't care about finding the first, last, shortest, or longest slice, we just want a slice. Example: from [2,3,-1,2,-4] we would like to find the slice [3,-1,2,-4], where 3 + (-1) + 2 (-4) = 0" 217

12.161. Find all the prime factors of a number entered 217

12.162. Find the largest prime factor of a number 217

12.163. You have a scooter which needs two tires. You are given three tires. Each tire has a max life of one year. What is the max time you can run your scooter? 217

12.164. If in a tree , a node can have more than 2 children , is it possible to traverse it in inorder ? If yes , then how would it be done ? 217

12.165. "What are the three types of basic variables in Perl? 217

12.166. What would you use to see if a file exists?" 217

12.167. What is "group by"? 217

12.168. Difference between inner join & outer join 217

12.169. Difference between final & finally 217

12.170. What is the significance of virtual destructor? 217

12.171. Discussion on virtual inheritance 217

12.172. "Different ways to pass parameters to a function (by value, by reference, by pointer). for the following cases. 217

12.173. What is the effect by keeping a constructor private? (in terms of inheritance) 218

12.174. Diference between deep copy & shallow copy 218

12.175. What are local static variables? Its usage & comparison with global static variables & class level static data members 218

12.176. Difference between Malloc vs Realloc 218

12.177. Given a regular expression and a text, find the min no of replacements to be done. 218

12.178. Check if there is a common node in 2 linked lists. 218

12.179. How to find min element in a stack in O(1), where the only operations are Push and Pop. 218

12.180. Swap two numbers without using a temporarily variable. 218

12.181. Design an algorithm and write code to find the common ancestor of two nodes in a tree 218

12.182. Describe what a singleton is and when you would use it. 218

12.183. write a program that accepts two mandatory arguments without using any built in date or time functions . The first argument is a string "[HH:MM {AM|PM}" and the second argument is an integer which denotes minutes. The minutes get added to the string. The return value or output of the program should be a string of the same format as the first argument. For example AddMinutes("10:23 AM", 13) would return "10:36 AM 218

12.184. How do you programitcally create & terminate a process on linux 218

12.185. What's the difference between mergesort and quicksort? When would you use each? 218

12.186. Implement the stack in such a way that push,pop and minimum find in the stack operations are o(1). 218

12.187. You have a large text file. Given any two words, find the shortest distance(in terms of number of words) between the two given words in the file. Can you make the searching operation in O(1) time? what about the space complexity for your solution?Ex:File contains:as was is the as the yahoo you me was the andWords given: the, wasAnswer should be: 0If Words given: you, theAnswer: 2 218

12.188. "Recently in one of my interview i faced a question with Cube.The question goes like this. Draw a cube,on each of the 8 nodes binary numbers is represented.(am helpless since, actually diagramatic presentation will make it sense) The interviewer now wants to know why or in what fashion he represents the nodes with the binary digits?" 219

12.189. Find the Mth last element of a singly linked list in the best possible time and most efficient use of memory. 219

12.190. Given a Binary Search Tree, find its depth. 219

12.191. You have a billion urls, where each has a huge page. How to you detect the duplicate documents? 219

12.192. Given a value and a binary search tree.Print all the paths(if there exists more than one) which sum up to that value. It can be any path in the tree. It doesn't have to be from the root. 219

12.193. Design a chat server 219

12.194. Describe TCP Hand-shake and optimize SQL. 219

12.195. What is a recursive mutex lock? 219

12.196. What's hashmap in STL? How do you use it? When would you use it? 219

12.197. What's the difference between a pointer and a reference? 219

12.198. Explain the following terms: virtual function, virtual class, pure virtual function 219

12.199. How do you distributed hash tables work? 219

12.200. Difference between a mutex and a semaphore. When can we use one but not the other? 219

12.201. How would you sort an array that consists of only zeros and ones in only one pass? 219

12.202. How do static variables differ in C and C++? 219

12.203. How would you bring the effect in c++ of static functions without using static? 219

12.204. What data structure did you use on your previous project? Why? How would you improve on that? 220

12.205. How would you bring the effect in C++ of static functions without using static 220

12.206. "Given a dictionary of millions of words, write a program to find the largest possible rectangle of letters such that every row forms a word (reading left to right) and every column forms a word (reading top to bottom). 220

12.207. Design a hash table to store phone #s. Your job is to write a hash function that has a parameter username, and generate a key. Username is unique, length 5 and can be A-Z, 0-9, space. Write a hash function that generate keys without collisions and use minimum memory. 220

12.208. Coding: Reverse linked list 220

12.209. How would the Destructor for Singleton look like ? 220

12.210. Use of virtual destructor 220

12.211. Practical usage of STL set. 220

13. Amazon 1- 10 Questions 220

13.1. Given a 2d matrix with characters and a dictionary. Find all the valid words in the 2d matrix. The words can be towards right, left , up or down. Exact code to be given. 220

13.2. Given a sorted array, write a function to search first occurrence of a number in the array. If not found return -1. 220

13.3. For a given BST, connect each of its right child to its inorder successor. 220

13.4. There is a binary tree of size N. All nodes are numbered between 1-N(inclusive). There is a N\*N integer matrix Arr[N][N], all elements are initialized to zero. So for all the nodes A and B, put Arr[A][B] = 1 if A is an ancestor of B (NOT just the immediate ancestor). 221

13.5. Given a file of n lines, where n-1 lines are identical and 1 line is different. Find the unique line in not more than one scan of the file. 221

13.6. There are exactly N advertising boards on the highway. Now a company want to advertise on some of these advertising boards (each advertising board costs some money). 221

13.7. How would you test each of the 3 layers in a 3-tier web app? 221

13.8. How would you test amazon search functionality on the home page? 221

13.9. Given an array with different numbers and a number of C,so how to find all the combinations which the sum is C..like..array={1,2,3,4},C=3,,return is 2,which contains two combinations{{1,2},{3}}. 221

13.10. You visit a website and it is slow today (it is not slow everyday). What could be the cause(s) of the slowness? 222

13.11. You have an array of binary numbers as ""00001101000001010100000...""... We need to find the First occurrence of 1 in this series.. using binary search. 222

13.12. Design a chess game. Write all classes and methods. 222

13.13. Given a large file of (x,y) coordinates. Find the k farthest points from origin. 222

13.14. Given a network of printers and systems. Allocate the nearest printer to each system. How will you handle dynamic addition of printers and systems. 222

13.15. Given a sorted array which is rotated n number of times. Find out how many times the array is rotated. Time complexity should be less than O(n). 222

13.16. Given a Binary Search tree along with the parent pointer, find the next largest node for the given node. Give the time and space complexity. The node Structure is 222

13.17. Given a string in the form of a Linked List, check whether the string is palindrome or not. Don’t use extra memory. Give the time complexity. The node structure is 223

13.18. given an array of charactes have to replace space with %20. where %20 is considered as 3 characters.write complete code to implement this. 223

13.19. Find the maxProduct of three numbers from a given integer array. 223

13.20. For 2 given array a[] and B[], find the highest index of A such that logical array A[0...i] and A[N-1...N-1-i] are same. 223

13.21. Given an array of integers such that each element is either +1 or -1 to its preceding element. Find 1st occurrence of a given number in that array without using linear search. 223

13.22. Function to reverse a c style sub-string 223

13.23. 1. N-Petrol bunk problem: There are n petrol bunks located in a circle. We have a truck which runs 1 km per 1 liter (mileage 1kmpl). Two arrays are given. The distances between petrol bunks are given in one array. Other array contains the no of liters available at each petrol bunk. We have to find the starting point such that if we start at that point , you we would able to visit entire circle without running out of fuel. Initially truck has no fuel 223

13.24. write a code to print the second largest element in a list 224

13.25. There is a HealthMonitor and two Servers (Primary and Secondary), all connected to one and another. 224

13.26. Write a class that will have following functions: 224

13.27. Write a class For Contacts on a device 224

13.28. Write a class for a parking garage: 224

13.29. Implement:1. a search that will return all the strings that match a sub-string 2. an insert into this datastructure 225

13.30. Implement an iterator for a Binary tree. It should have the following things: 225

13.31. If a function is given mostCommonChar(String str, int num) , 225

13.32. Write a C program to find the number of shift required to convert one string to another. Check all the corner cases. 225

13.33. Adding Very Large Numbers. Write clean code for it. please check all corner cases.. 225

13.34. Design a singleton design pattern. 225

13.35. Design a class in C++ such that only one object of it can be created. 226

13.36. Find the 3rd closest element in a bst.You will be given a pointer to root and a value within the tree against which the closest has to be figured out. (closeness is in terms of value, not by distance ) and then follow up qn: for finding the kth closest in a bst. 226

13.37. A video streaming server is generating the following data. Find the potential customers facing buffering issues. 226

13.38. Design an online hotel reservation system. 226

13.39. Design a furniture store with Tables and chairs. Write a constructor for chair and table 226

13.40. You are given a UNIX path with dot (current) and two dots (parent). Convert this to an absolute path 226

13.41. Input is given a binarytree and out is sum of the all the children data and its node data . 227

13.42. Given a sorted array consisting 0's and 1's. find the index of first '1'. write a complete program which takes less time complexity. and test all boundary conditions also. 227

13.43. Explain Collaborative Caching? 227

13.44. There are lots of string in a file. Find the longest string that could be made from the other strings in the file. 227

13.45. Given an array A of length n where each element is 1..k where k is much smaller than n, 227

13.46. Code to create a file system.... Have classes like directory, file and all 227

13.47. In a BST, I want to replace all nodes with value which is the sum of all the nodes which are greater than equal to the current node. 228

13.48. You are given a BST, and min, max elements. Your task is to trim this BST so that it contains the elements between the min and the max elements. 228

13.49. Design a DS for storing browsing history. 228

13.50. Design a Calculator. Details about the class variables, datastructure to be used etc. 228

13.51. Generate a solution when multiple threads want to just read in their critical sections and when nobody is writing in the critical section. 229

13.52. Finding border of a binary tree.Given a Binary tree print all the nodes that form the boundary. 229

13.53. For given N\* N matrix, 229

13.54. how do you handle your thread, to avoid dead lock and efficient(generally question) 229

13.55. what is Materialized view, is any different from View. 229

13.56. Write a program to swap kth node from first and kth node from last in a linked list . 229

13.57. Design a MMORPG game in internet scale. Assume only available action for the players is watch and move. 229

13.58. Given a matrix that contains 0s and 1s, find the shortest exit and print the path. You can navigate in top,bottom,left or right directions. 229

13.59. In a stream of numbers, keep track of 1 million max numbers. 230

13.60. find the Langford sequence for a given N if it exists? Details of Langford sequence - https://en.wikipedia.org/wiki/Langford\_pairing 230

13.61. Given an array of integers,write a function that retrieves unique instances of any duplicates, returning them in a new array - 230

13.62. In a shop, product X is available in different quantities q1,q2,q3...... with price tags p1,p2,p3,... 230

13.63. Wap to find kth largest element in a binary search tree 230

13.64. Given a binary tree convert it to doubly linked list, with left pointer of binary tree as prev pointer of doubly linked list and right pointer of binary tree as next pointer of doubly linked list. 230

13.65. Given a linked list, print n nodes from tail of the list in reverse order 231

13.66. How can you implement a Hashtable with any given data structure 231

13.67. Given an integer array, return the combinations of 4 array values whose sum is x 231

13.68. Create a Session Manager class that iterates thru Session objects throw stale sessions out - How would you automatically purge Session objects that have not been active for 30 seconds or (n) seconds? The answer needs to handle millions of sessions. 231

13.69. Youtube is not playing the selected video. What could be the problem and how to debug this issue? 231

13.70. Write a program to check whether a substring is present in a main string. 232

13.71. Write all possible test cases for SMS (Short Messaging Service) on a mobile device. 232

13.72. Data Structure for node of linked list 232

13.73. Design an algorithm to remove the duplicate characters in a string without using any additional buffer. NOTE: One or two additional variables are fine. An extra copy of the array is not. also do with o(n2),o(n),o(1) ,write test case and breake the algorithm whatever you write. 232

13.74. Length is given as input.Print all possible permutations of numbers between 0-9. 232

13.75. Create a basic implementation of Deferred. 232

13.76. Using the mythical Hydra as an example, create a button that is destroyed by clicking it, but two new buttons are created in it's place. 233

13.77. Create a function that will reverse the words in a sentence. 233

13.78. You have unique ASCII characters. How you can sort them ? 233

13.79. N boys are sitting in a circle. Each of them have some apples in their hand. You find that the total number of the apples can be divided by N. 233

13.80. You are given with space of binary codewords, and you have to come up with an algo to generate all subspace of size 2^1 , 2^2 ,2^3 . . .2^n of that set. 233

13.81. given an array A[0-n], find the combination of A[i] and A[j] such that 234

13.82. Reverse words in a sentence. 234

13.83. How can you implement queue? 234

13.84. Write an algorithm that will take two dates and tell you if they are more than a month apart, less than a month apart or exactly a month apart. 234

13.85. Some theory 234

13.86. There are different buildings in one environment, each with machines that can handle one request at a time. How would you design the request handling so that there is no single fail-point and is scalable. 234

13.87. Given a set of array of size n, return all possible subset of size k. 235

13.88. Given a sequence of non-negative integers find a subsequence of length 3 having maximum product with the numbers of the subsequence being in ascending order. 235

13.89. Special Property Numbers: 235

13.90. This is on Additive Number Property 235

13.91. You are given a 2-D array with same number of rows and columns. You have to determine the longest snake in the array. The property to find the snake is the difference between the adjacent(left, right, up or down) should be either 1 or -1. If there are more than one snakes with maximum length, the output should print both of them. 235

13.92. Class and Data Structure Design for a ""metric"" system to determine the top song of a band. Two Web Service calls: 236

13.93. Design a Text Editor, in term of class diagram and data structure required to store the data/text and Insert, Delete, search in both direction and Edit operation. Calculate the time complexity for all operation. 236

13.94. Design a Cache System, with the appropriate data structure and operation with the time complexity. 236

13.95. Given an array find the next greatest element to the right of it. [4, 5, 2, 25] 236

13.96. I was given a space of binary codewords containing 2^k codewords of word length n. And I was asked to generate all possible subspace of size 2^1, 2^2, 2^3 . . . . .,2^k. 237

13.97. Given an array of integers, find Pythagorean triplets. 237

13.98. Given a Directed graph, and a source point say S, and say C is the most distant node from S, you need to find the route from S to C . Suppose ,for example there is a path from A-> B- >C, and a path from A->B->D->C, here the most distant node is C with path length as 4. Also, each edge has unit length. Since the graph is directed , you need to implement an algo which takes care of the cycles in the graph. How would you implement this? 237

13.99. Given a string, you need to find super string by word match. i.e. all words in the input string has to occure in any order in output string. 237

13.100. there is a log file which contains info in below format: 238

13.101. Consider a city (visualize a circle). It has n petrol stations in it. You are given the maximum amount of petrol that can be filled at each of these stations. You are also given the distance between one station to the next one. The aim is to cover the entire city and come back to the start point. Assume that 1 liter of petrol will last for 1km. 238

13.102. You are given an integer array, where all numbers except for TWO numbers appear even number of times. Find out the two numbers which appear odd number of times. 238

13.103. int board[8][8] each value in the matrix represents a character. 1-9 number represents all whites and 11-19 represents all blacks. 238

13.104. The cost of a stock on each day is given in an array, find the max profit that you can make by buying and selling in those days 238

13.105. A = {5, 3, 8, 9, 16} After one iteration A = {3-5,8-3,9-8,16-9}={-2,5,1,7} After second iteration A = {5-(-2),1-5,7-1} sum =7+(-4)+6=9 Given an array, return sum after n iterations 239

13.106. You have a dictionary, D, that stores the positions of words in a document by mapping words (strings) to positions in the document (arrays of ints.) 239

13.107. Write a function that finds out if any two numbers within that array add up to a target. 239

13.108. Given a double ended queue with front and rear as two pointers, if a two people play a game. Say removal of number from a given list- from either side of the list. given the fact that the players can view the list of numbers, maximize the chance of each player winning? 239

13.109. There are n coins in a line. (Assume n is even). Two players take turns to take a coin from one of the ends of the line until there are no more coins left. The player with the larger amount of money wins. 239

13.110. Find the next greatest number that can be form by same digit of the given number. sample test case input 4765 output 5467 243

13.111. waf to swap kth node from first with kth node form last 243

13.112. WAP to find the longest arithmetic sequence in given array. Return number of element in series. 243

14. Preparation 287

15. C++ Notes 289

15.1. Complete Reference Notes 289

15.2. Simulating final class in C++ 298

16. Effective C++ 299

16.1. Prefer consts, enums, and inlines to #defines 299

16.2. Use const whenever possible 301

16.3. Item 4: Make sure that objects are initialized before they're used 303

16.4. Know what functions C++ silently writes and calls 304

16.5. Explicitly disallow the use of compiler-generated functions you do not want 305

16.6. Declare destructors virtual in polymorphic base classes 306

17. Important Data structures 306

17.1. Tries 306

17.2. Radix Trees 307

17.3. B-Tree 307

17.4. Suffix Tree 314

17.5. Prefix tree 314

17.6. B+ Trees 314

17.7. AVL 329

17.8. Red Black Tree 335

17.9. Stable Marriage Problem 335

17.10. Stable Roommate problem 335

18. Memory Organization of a program 335

19. General Questions answered 345

20. Computer Algorithms 348

21. Windows API 349

21.1. Synchronization Functions 349

21.1.1. Asynchronous functions 350

21.1.2. Condition variable and SRW lock functions 350

21.1.3. Critical section functions 351

21.1.4. Event functions 351

21.1.5. One-time initialization functions 351

21.1.6. Interlocked Functions 351

21.1.7. Mutex functions 360

21.1.8. Private namespace functions 360

21.1.9. Semaphore functions 360

21.1.10. Singly-linked list functions 360

21.1.11. Synchronization barrier functions 361

21.1.12. Timer-queue timer functions 361

21.1.13. Wait functions 361

21.1.14. Waitable-timer functions 362

21.2. Pipes 362

# Arrays and Strings

## Find the elements common to two integer arrays

Sort and move two pointers

def common\_elements(a, b):

a.sort()

b.sort()

i, j = 0, 0

common = []

while i < len(a) and j < len(b):

if a[i] == b[j]:

common.append(a[i])

i += 1

j += 1

elif a[i] < b[j]:

i += 1

else:

j += 1

return common

print 'Common values:', ', '.join(map(str, common\_elements([1, 2, 4, 8], [1, 4, 9])))

Hash and search

import java.util.\*;

public class CountTest {

public static void main(String... args) {

Integer[] array1 = {9, 4, 6, 2, 10, 10};

Integer[] array2 = {14, 3, 6, 9, 10, 15, 17, 9};

**Set hashSet = new HashSet(Arrays.asList(array1));**

Set commonElements = new HashSet();

for (int i = 0; i < array2.length; i++) {

if (hashSet.contains(array2[i])) {

commonElements.add(array2[i]);

}

}

System.out.println("Common elements " + commonElements);

}

}

For very large arrays

Assuming integer size is 4 bytes. Now we can have maximum of 2^32 integers i.e I can have a bitvector of 2^32 bits (512 MB) to represent all integers where each bit reperesents 1 integer. 1. Initialize this vector with all zeroes 2. Now go through one file and set bits in this vector to 1 if you find an integer. 3. Now go through other file and look for any set bit in bit Vector.

Time complexity O(n+m) space complexity 512 MB

## Given an array of size n. It contains numbers in the range 1 to n. Each number is present at least once except for 2 numbers. Find the missing numbers.

* + One way is to use bitsets to store the numbers that are present in the array and keep on checking if the bit is already set for duplicate numbers.
  + In the end of process look for 0 bits in the numbers and these are the missing numbers
  + Second approach is to use the array itself and use the array elements as indices to negate the integers found. IF the element is already negated we have a duplicate number and in the end the indices that have +ive numbers are the missing ones.
* <http://discuss.joelonsoftware.com/default.asp?interview.11.778142.8>

## Given an array of size n, containing every element from 1 to n+1, except one. Find the missing element.

Sum all the elements and subtract it from the sum of numbers 1 to n+1.

## Given an array of size n. It contains numbers in the range 1 to n. Each number is present at least once except for 1 number. Find the missing number.

Ques: What is there in place of missing number?

1. A number from 1 – n repeats itself. NEXT problem…

2. zero. See previous problem…

3. Number out of range from 1-n. Sum all the elements except for the odd one use 0.

## Find a missing and repeating integer from an array of N integers ranging from 1 to N.

//let me try

**int** findDup (**int** \*a, **int** len) { // len is 1001 here

**int** found, i, t;

**for** (i = 0; i < len; i++) {

t = a[i] > 0 ? a[i] : -a[i];

**if** (a[t] > 0)

a[t] = -a[t]; // use negative as a mark

**else** { // if it is already marked, it must be a dup!

found = t;

**break**;

}

}

**for** (i = 0; i < len; i++)

a[i] = (a[i] > 0) ? a[i] : -a[i]; // restore the array

**return** found;

}

To get the missing number keep on adding the numbers of array in for loop. Let S be the sum.

Missing number = (Sum of numbers 1..N) – (Sum of array elements – dup)

Missing number = (N\*(N+1)/2) – (S – t)

## You have an array in which every number is repeated odd number of times except one, you have to give that with O(n) time complexity

Traverse through the array, hash the value if it is not present and delete the hash table entry if it is already hashed. By the time we process the array, only the odd no items will be hashed, then traverse the array to find out which was not hashed. That was our answer...

Space: O(no of distinct elements in array), worst case O(n)

Running time: O(n)

## Find the two numbers with odd occurrences in an unsorted array

Given an unsorted array that contains even number of occurrences for all numbers except two numbers. Find the two numbers which have odd occurrences in O(n) time complexity and O(1) extra space.

Examples:

Input: {12, 23, 34, 12, 12, 23, 12, 45}

Output: 34 and 45

Input: {4, 4, 100, 5000, 4, 4, 4, 4, 100, 100}

Output: 100 and 5000

Input: {10, 20}

Output: 10 and 20

A naive method to solve this problem is to run two nested loops. The outer loop picks an element and the inner loop counts the number of occurrences of the picked element. If the count of occurrences is odd then print the number. The time complexity of this method is O(n^2).

We can use sorting to get the odd occurring numbers in O(nLogn) time. First sort the numbers using an O(nLogn) sorting algorithm like Merge Sort, Heap Sort.. etc. Once the array is sorted, all we need to do is a linear scan of the array and print the odd occurring number.

We can also use hashing. Create an empty hash table which will have elements and their counts. Pick all elements of input array one by one. Look for the picked element in hash table. If the element is found in hash table, increment its count in table. If the element is not found, then enter it in hash table with count as 1. After all elements are entered in hash table, scan the hash table and print elements with odd count. This approach may take O(n) time on average, but it requires O(n) extra space.

**A O(n) time and O(1) extra space solution:**

Let the two odd occurring numbers be x and y. We use bitwise XOR to get x and y. The first step is to do XOR of all elements present in array. XOR of all elements gives us XOR of x and y because of the following properties of XOR operation.

1) XOR of any number n with itself gives us 0, i.e., n ^ n = 0

2) XOR of any number n with 0 gives us n, i.e., n ^ 0 = n

3) XOR is cumulative and associative.

So we have XOR of x and y after the first step. Let the value of XOR be xor2. Every set bit in xor2 indicates that the corresponding bits in x and y have values different from each other. For example, if x = 6 (0110) and y is 15 (1111), then xor2 will be (1001), the two set bits in xor2 indicate that the corresponding bits in x and y are different. In the second step, we pick a set bit of xor2 and divide array elements in two groups. Both x and y will go to different groups. In the following code, the rightmost set bit of xor2 is picked as it is easy to get rightmost set bit of a number. If we do XOR of all those elements of array which have the corresponding bit set (or 1), then we get the first odd number. And if we do XOR of all those elements which have the corresponding bit 0, then we get the other odd occurring number. This step works because of the same properties of XOR. All the occurrences of a number will go in same set. XOR of all occurrences of a number which occur even number number of times will result in 0 in its set. And the xor of a set will be one of the odd occurring elements.

// Program to find the two odd occurring elements

#include<stdio.h>

/\* Prints two numbers that occur odd number of times. The

function assumes that the array size is at least 2 and

there are exactly two numbers occurring odd number of times. \*/

void printTwoOdd(int arr[], int size)

{

int xor2 = arr[0]; /\* Will hold XOR of two odd occurring elements \*/

int set\_bit\_no; /\* Will have only single set bit of xor2 \*/

int i;

int n = size - 2;

int x = 0, y = 0;

/\* Get the xor of all elements in arr[]. The xor will basically

be xor of two odd occurring elements \*/

for(i = 1; i < size; i++)

xor2 = xor2 ^ arr[i];

/\* Get one set bit in the xor2. We get rightmost set bit

in the following line as it is easy to get \*/

set\_bit\_no = xor2 & ~(xor2-1);

/\* Now divide elements in two sets:

1) The elements having the corresponding bit as 1.

2) The elements having the corresponding bit as 0. \*/

for(i = 0; i < size; i++)

{

/\* XOR of first set is finally going to hold one odd

occurring number x \*/

if(arr[i] & set\_bit\_no)

x = x ^ arr[i];

/\* XOR of second set is finally going to hold the other

odd occurring number y \*/

else

y = y ^ arr[i];

}

printf("\n The two ODD elements are %d & %d ", x, y);

}

/\* Driver program to test above function \*/

int main()

{

int arr[] = {4, 2, 4, 5, 2, 3, 3, 1};

int arr\_size = sizeof(arr)/sizeof(arr[0]);

printTwoOdd(arr, arr\_size);

getchar();

return 0;

}

Output:

The two ODD elements are 5 & 1

Time Complexity: O(n)

Auxiliary Space: O(1)

## Suppose we have an array like 1,2,3,4,5,a,b,c,d,e where we have always even number of elements. First half of the elements are integers and second half are alphabets we have to change it to like 1,a,2,b,3,c,4,d,5,e in place i.e no use of any extra space, variables are allowed ..

int newSortArr(int \*arr, int nLength)

{

/\*

1 2 3 4 5 a b c d e

1 a b c d 2 3 4 5 e

1 a 2 3 4 b c d 5 e

1 a 2 b c 3 4 d 5 e

1 a 2 b 3 c 4 d 5 e

\*/

if (NULL == arr)

return 0;

int i = 1;

int j = nLength-2;

int mid = nLength>>1;

while (i < j)

{

for (int m = i;m < mid;m++)

swap(arr+m,arr+mid+m-i);

i++;

j--;

}

return 1;

}

## Write a function that given two strings s1 and s2, it will return s1 minus s2, and it has to use same s1 location. ex: s1 = "abcdB" ,s2 = "b" ,s1 Minus s2 = "acdB"

Assumptions:

1. S2 as a whole is to be subtracted.

2. All occurrences of S2 are subtracted.

# include <conio.h>

# include <stdio.h>

#include <string.h>

**int** main ()

{

**char** s1[]="sanjay kumar", s2[]="a";

**char** \*s3; **int** i,len;

**while** (1)

{

s3=strstr(s1,s2);

**if**(s3==NULL)

{

**break**;

}

strncpy(s3,s3+strlen(s2),strlen(s3+strlen(s2)));

len=strlen(s1)-strlen(s2);

**for**(i=strlen(s1);i>=len;i--)

s1[i]='\0';

}

printf("%s",s1);

getch();

**return** 0;

}

## Given an crypted array obtain the original text . Should be implemented without extra space . Eg: Crypt array : a3b4c3 decrypt array : aaabbbbccc Imagine the array contains sufficient memory to hold the decrypt ?

First get rid of the 0,1,2 cases. Ex a3b1b2c3🡪 a3bbbc3

Then start putting the chars from end of the array.

## Given a string, compress using run-length encoding. Example: Input: aaabbac Output: a3b2ac

#include <sstream>

string encodeRLE(const string &src)

{

size\_t len = src.length();

ostringstream os;

for(size\_t i = 0; i < len; ) {

size\_t j;

int count = 1;

for(j = i + 1; j < len; j++) {

if(src[i] == src[j])

++count;

else

break;

}

if(count != 1)

os << src[i] << count;

else

os << src[i];

i = j;

}

return os.str();

}

## In a nxn matrix, data provided like below. We need to find the groups of 1s with the adjactent column and row.

eg)

0 0 0 0

1 1 1 1

0 0 0 0

group of 1 is 1

1 0 0 0

0 0 0 1

1 1 0 0

group of 1s is 3

Any thought how to get the set of groups.

Sol:

input array (a) and flag array are the members of class. Below is the code for counting both zero groups and one groups.

**private** **void** CountOneGroups()

{

flag = **new** **boolean**[a.length][a[0].length];

**for**(**int** i = 0; i < a.length; i++)

**for**(**int** j = 0; j < a[i].length; j++)

flag[i][j] = **false**;

**int** zeroCount, oneCount;

zeroCount = 0;

oneCount = 0;

**for**(**int** i = 0; i < a.length; i++)

**for**(**int** j = 0; j < a[i].length; j++)

**if**(flag[i][j] != **true**)

{

**if**(a[i][j] == 0)

zeroCount++;

**else**

oneCount++;

FourConnected(i, j, a[i][j]);

}

System.*out*.println("Zero Group Count : " + zeroCount);

System.*out*.println("One Group Count : " + oneCount);

}

**private** **void** FourConnected(**int** i, **int** j, **int** value)

{

**if**(!flag[i][j])

{

flag[i][j] = **true**;

**if**(i - 1 >= 0 && a[i - 1][j] == value)

FourConnected(i - 1, j, value);

**if**(j - 1 >= 0 && a[i][j - 1] == value)

FourConnected(i, j - 1, value);

**if**(i + 1 < a.length && a[i + 1][j] == value)

FourConnected(i + 1, j, value);

**if**(j + 1 < a[0].length && a[i][j + 1] == value)

FourConnected(i, j + 1, value);

}

}

## Write an algorithm to find an element in a 2D array such that the element should be maximum in the row and minimum in the column?

Calculate the maximum in a row, in the loop and save its indexes(imax,jmax).

now, find in column jmax lowest element by iterating over i. only one element is required so no extra space required.

**for**(i=0;i<row;i++)

{

min = INT\_MAX;

max = INT\_MIN;

imax = jmax = -1;

**for**(j=0;j<col;j++)

{

**if**(a[i][j]>max) {imax = i; jmax = j; max = a[i][j];}

}

**for**(j=0;j<row;j++)

{

**if**(a[j][jmax]<min) { imin = j; min = a[j][jmax];}

}

**if**(imin == imax){printf("found at %d %d",imax,jmax)**return**;}

}

Sol2:

1. for-loop every row, find the maximum value of the row, and save its index into an array, e.g: row[i] = index of maximum

2. for-loop every column, find the minimum value of the column, and save its index into another array. e.g: col[j] = index of minimum.

3. Check row[i], col[j], i,j, if we can find a pair, row[i]=j, and col[j]=i, then the m[j][i] is the result.

for example: matrix m[4][4] as following

3, 5, 9, 4

1, 10,11,6

8, 7, 14,13

12,2,16,15

row[0]=2, col[0]=1,

row[1]=2, col[1]=3,

row[2]=2, col[2]=0,

row[3]=2, col[3]=0,

row[0]=2 and col[2]=0, so the result is m[0][2] is the result.

for example2: change the m[0][2] from 9 to 19, the col[2] = 1.

so, this time the answer is row[1]=2 and col[2]=1, m[1][2]

Time is O(n^2), Space is O(c)

## How to rotate the array with o(n) or o(nlogn). eg) A[]={A,B,C,D,E} rotate Index – 2 It should be {C,D,E,A,B}

Group 1: A B  
Group 2: C D E  
{A B C D E} = { Group 1, Group 2}  
Reverse Group 1 = {B, A}  
Reverse Group 2 = {E, D, C}  
Reverse {Group 1', Group 2'} = {C, D, E, A, B}

**void** reverse(vector<**int**> &v, **int** start, **int** end) {

**if**(v.size() < (end - start)) **return**;

**for**(;start < end; start++, end--) {

std::swap(v[start], v[end]);

}

}

**void** rotate(vector<**int**> &v, **int** k) {

reverse(v, 0, k-1);

reverse(v, k, v.size() - 1);

reverse(v, 0, v.size() - 1);

}

## Given an array and a number K. You have to find out longest subset from the array whose all pair sum will be greater than k. ex: {8,3,4,1,6,2,5,7,9} and K=12 ans: {8,6,7,9} or {5,7,8,9}

First we sort the array. Assume A is sorted.

Let B be a subset of A whose minimum element is b. Now every element x in B must be at least K-b.

So, here is the algorithm: for every element in A, see how many elements are at least K minus that element. Take the maximum over all.

This takes O(n logn) in total

sort(A)

max = 0;

max\_index = 0;

**for** (**int** i=0; i < A.size(); i++){

**int** x = {the minimum index such that} A[x] >= K-A[i]

**if** (n-x >= max){

max = n-x;

max\_index = i;

}

}

**for** (**int** i=0; i < A.size(); i++)

**if** (A[i] >= A[max\_index])

cout << A[i] << " ";

## You have two arrays A and B of strings. In the array B all element are from A except one. You have find out the string which is extra in B in O(n) time. ex:

A = {"abc", "bcd", "dpr"};

B = {"abc", "mnp", "bcd", "dpr"};

You have find out the string which is extra in B in O(n) time.

In the above example it is "mnp".

1. Hash A and lookup B
2. Represent each string as number and xor them.

**Ques: Why to use 50 for subtracting. Better would be convert char to hex-numbers**

I guess this solution will work.. in O(1) space and O(m) times // m is length of array

// Assuming only AlphaNumeric characters allowed.

for every character in the string, replace it with (ascii -50). this will generate a number. take the XOR and proceed further.

In the final output of XOR for every 2 digits add 50 to it and replace with the correct character of that ascii.

for string "abc" numeric representation will be 474849.

for string "mnp" numeric representation will be 596061.

calculate numeric representation for all the string as and when u traverse the array and take bitwise XOR.

in the above example output will be 596062 in numeric format. Output in string format characters will ascii(59+50)(60+50)(62+50) == mnp

## Given a m\*n matrix and a person is sitting in (0,0) box, and he has to go to the (m-1,n-1) box of the matrix .And the person can only go to right or down box from its current box position . We need to find out the number of ways he can reach from start to destination box .

The solution can be generated by looking at the fact that we have to travel total of n-1 distance to right and m-1 distance downwards. So we have total of m+n-2 steps. Now out of these we need to select the m-1 towards downward steps which can be done in m+n-2Cm-1 ways

## Given 2 dimensional sorted array (Both row and column wise sorted) write an efficient code to find median.

1 3 4 8 9

2 5 18 25 50

6 7 22 45 55

take the middle element, here 18, all element before that in the left and above will be smaller than 18, and all element on the right, and bottom will be larger than 18. Thus we need to find median of the remaining elements i.e. {6, 7}, 18, {8, 9}. We can sort these sorted array sub arrays to get median. This will work if we have odd number of row and columns, in case of even rows and/or columns, we shall have modify logic a little, but idea will remain same. The complexity of this approach will be log(max(row, col)), where matrix is of dimension row X column

## You are given an array of integers, say array1 of size n. You have to create another array (say array2) and fill its contents by following this rule: array2[i] will hold the value of the left-most integer from the range array1[i+1] to array1[n-1] such that it is greater than array1[i]. If no such element exists, assign -1.

Sol:

**public** **int**[] LeftmostGreater(**int** a[])

{

**int** b[] = **new** **int**[a.length];

**int** i = 0;

**int** j = 1;

**while**(i<a.length)

{

**for**(; j < a.length;j++)

{

**if**(a[i] < a[j])

{

**for**(; i < j; i++)

{

b[i] = a[j];

}

}

}

**for**(;i<j;i++)

b[i] = -1;

}

**return** b;

}

## Given an array with positive and negative numbers find the first continuous subarray that sums to 0.

Given an int[] input array, you can create an int[] tmp array where tmp[i] = tmp[i - 1] + input[i]; Each element of tmp will store the sum of the input up to that element.

Now if you check tmp, you'll notice that there might be values that are equal to each other. Let's say that this values are at indexes j an k with j < k, then the sum of the input till j is equal to the sum till k and this means that the sum of the portion of the array between j and k is 0! Specifically the 0 sum subarray will be from index j + 1 to k.

NOTE: if j + 1 == k, then k is 0 and that's it! ;)

NOTE: The algorithm should consider a virtual tmp[-1] = 0;

NOTE: An empty array has sum 0 and it's minimal and this special case should be brought up as well in an interview. Then the interviewer will say that doesn't count but that's another problem! ;)

The implementation can be done in different ways including using a HashMap with pairs but be careful with the special case in the NOTE section above.

Example:

int[] input = {4, 6, 3, -9, -5, 1, 3, 0, 2}

int[] tmp = {4, 10, 13, 4, -1, 0, 3, 3, 5}

Value 4 in tmp at index 0 and 3 ==> sum tmp 1 to 3 = 0, length (3 - 1) + 1 = 4

Value 0 in tmp at index 5 ==> sum tmp 0 to 5 = 0, length (5 - 0) + 1 = 6

Value 3 in tmp at index 6 and 7 ==> sum tmp 7 to 7 = 0, length (7 - 7) + 1 = 1

**public** **int**[] FirstSubarraySumZero(**int** a[])

{

HashMap<Integer,Integer> hashMap = **new** HashMap<Integer,Integer>();

**int** sum = 0;

**int** result[] = {-1,-1};

**for**(**int** k=0;k<a.length;k++)

{

sum += a[k];

**if**(sum == 0)

{

result[0] = 0;

result[1] = k;

**return** result;

}

**if**(hashMap.containsKey(sum))

{

result[0] = hashMap.get(sum)+1;

result[1] = k;

**return** result;

}

hashMap.put(sum, k);

}

**return** result;

}

## You have 2 character arrays. The arrays have characters ranging from a-z (all small letters).1. Merge the two arrays, 2. Sort the array. Perform the above in O(N)

## Given two sorted arrays. A and B. there are empty spaces in A which is same size of B. Merge them.

## Sorted Array Merge Problem: Design an algorithm to merge two sorted positive integer arrays A and B. Size of array A is n and size of array B is m. Array A has m empty blocks(-1) at random places. Merge array B into A such that array A is sorted and does not have empty blocks.

## Write two algorithms to remove duplicates from an integer array.

1. One method is sort the array O(nlogn) & remove duplicates O(n)

Original sequence will not be preserved but no exta space.

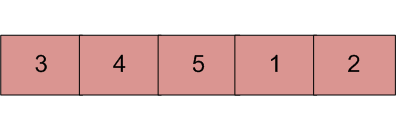
2. create hash map

Original sequence will be preserved but exta space is required.

## The sorted array is rotated by some factor. And we need to search a key? O(logn) ?

<http://www.geeksforgeeks.org/search-an-element-in-a-sorted-and-pivoted-array/>

An element in a sorted array can be found in O(log n) time via binary search. But suppose I rotate the sorted array at some pivot unknown to you beforehand. So for instance, 1 2 3 4 5 might become 3 4 5 1 2. Devise a way to find an element in the rotated array in O(log n) time.



**Solution:**  
Thanks to Ajay Mishra for initial solution.

**Algorithm:**  
Find the pivot point, divide the array in two sub-arrays and call binary search.  
The main idea for finding pivot is – for a sorted (in increasing order) and pivoted array, pivot element is the only only element for which next element to it is smaller than it.  
Using above criteria and binary search methodology we can get pivot element in O(logn) time

Input arr[] = {3, 4, 5, 1, 2}

Element to Search = 1

1) Find out pivot point and divide the array in two

sub-arrays. (pivot = 2) /\*Index of 5\*/

2) Now call binary search for one of the two sub-arrays.

(a) **If** element is greater than 0th element then

search in left array

(b) **Else** Search in right array

(1 will go in else as 1 < 0th element(3))

3) **If** element is found in selected sub-array then return index

**Else** return -1.

**Implementation:**

/\* Program to search an element in a sorted and pivoted array\*/

#include <stdio.h>

int findPivot(int[], int, int);

int binarySearch(int[], int, int, int);

/\* Searches an element no in a pivoted sorted array arrp[]

of size arr\_size \*/

int pivotedBinarySearch(int arr[], int arr\_size, int no)

{

int pivot = findPivot(arr, 0, arr\_size-1);

// If we didn't find a pivot, then array is not rotated at all

if (pivot == -1)

return binarySearch(arr, 0, arr\_size-1, no);

// If we found a pivot, then first compare with pivot and then

// search in two subarrays around pivot

if (arr[pivot] == no)

return pivot;

if (arr[0] <= no)

return binarySearch(arr, 0, pivot-1, no);

else

return binarySearch(arr, pivot+1, arr\_size-1, no);

}

/\* Function to get pivot. For array 3, 4, 5, 6, 1, 2

it will return 3 \*/

int findPivot(int arr[], int low, int high)

{

if (high < low)

return -1;

int mid = (low + high)/2; /\*low + (high - low)/2;\*/

if (mid < high && arr[mid] > arr[mid + 1])

return mid;

if (arr[low] >= arr[mid])

return findPivot(arr, low, mid-1);

else

return findPivot(arr, mid + 1, high);

}

/\* Standard Binary Search function\*/

int binarySearch(int arr[], int low, int high, int no)

{

if (high < low)

return -1;

int mid = (low + high)/2; /\*low + (high - low)/2;\*/

if (no == arr[mid])

return mid;

if (no > arr[mid])

return binarySearch(arr, (mid + 1), high, no);

else

return binarySearch(arr, low, (mid -1), no);

}

/\* Driver program to check above functions \*/

int main()

{

int arr[] = {3, 4, 5, 1, 2};

int arr\_size = sizeof(arr)/sizeof(arr[0]);

int no = 3;

printf("Index of the element is %d", pivotedBinarySearch(arr, arr\_size, no));

return 0;

}

Output:

Index of the element is 0

Please note that the solution may not work for cases where the input array has duplicates.

**Time Complexity** O(logn)

## Given a rotated-sorted array find the point of rotation. Complexity less than O(n).

Example:

Index: 0 1 2 3 4

Value: 4 5 1 2 3

Ans: 2 (index)

int ModifiedBinarySarch(int a[],int n)

{

int start=0;

int end=n-1;

int mid;

while(start<end)

{

mid=(start+end)/2;

if(a[mid]<a[mid-1])

break;

if(a[mid]>a[start])

{

start=mid+1;

}

else

{

end=mid-1;

}

}

return mid;

}

int find(int a[], int s, int e)

{

if((e-s)==1)

return e;

int m;

m=(s+e)/2;

if((a[e]-a[m])>0 && (a[m-1]-a[s])>=0)

return m;

if((a[e]-a[m])>0)

return find(a, s, m-1);

else

return find(a,m,e);

}

int main()

{

int a[7]={13,24,35,46,0,4,12};

cout<< find(a,0,6);

return 0;

}

## Given a matrix, print it spirally.

Input:

1 2 3

4 5 6

7 8 9

Output: 1 2 3 6 9 8 7 4 5

**public** **void** printSpiral(**int** arr[][], **int** row, **int** col, **int** m, **int** n) {

**if** (row > m && col > n)

**return**;

//This applies to square matrix m==n, with odd rows and cols . and print the last middle element

**if** (m == n && row == m)

System.out.print(arr[row][col]);

**int** i, j;

**for** (i = row, j = col; j < n; j++)

System.out.print(arr[i][j] + " ");

**for** (i = row, j = n; i < m; i++)

System.out.print(arr[i][j] + " ");

**for** (i = m, j = n; j > col; j--)

System.out.print(arr[i][j] + " ");

**for** (i = m, j = col; i > row; i--)

System.out.print(arr[i][j] + " ");

printSpiral(arr, row + 1, col + 1, m - 1, n - 1);

**return**;

}

**int**[][] arr = { {1,2,3,4},{21,22,23,24}, {14,15,16,17}};

solution.printSpiral(arr, 0, 0, 2, 3);

## There is a matrix where the cost of moving horizontally is 1, vertically is 1 and diagonally is 1.1. Now given two points, what is the number of shortest paths between these two points?

Let the two points be (A,B) and (C,D)

(Max(Length,Breadth) - DiagonalMoves) + 1.1\*DiagonalMoves

=> MAX(|A-C|, |B-D|) - MIN(|A-C|,|B-D|) + 1.1\* MIN(|A-C|, |B-D|)

=> MAX(|A-C|, |B-D|) + 0.1 \* MIN(|A-C|, |B-D|)

## Find third largest element of an array in a single pass?

Kth order statistic problem

Sol1 : Keep three variables. and keep on updating them. O(3n)

Sol2: Partition and quickselect.

partion the array on some pivot element if the element is at Kth location we have found it.

If the poivot is at location < K, take the second half and look for K- pivot location.

If pivot lands in > K, take first half and look for Kth element.

function partition(list, left, right, pivotIndex)

pivotValue := list[pivotIndex]

swap list[pivotIndex] and list[right] // Move pivot to end

storeIndex := left

for i from left to right - 1

if list[i] < pivotValue

swap list[storeIndex] and list[i]

increment storeIndex

swap list[right] and list[storeIndex] // Move pivot to its final place

return storeIndex

function select(list, left, right, k)

loop

select pivotIndex between left and right

pivotNewIndex := partition(list, left, right, pivotIndex)

pivotDist := pivotNewIndex - left + 1

if pivotDist = k

return list[pivotNewIndex]

else if k < pivotDist

right := pivotNewIndex - 1

else

k := k - pivotDist

left := pivotNewIndex + 1

## Given a 4GB file of numbers. You are asked to find the product of K largest numbers. The size of the main memory is 1 GB. Give an efficient method to find.

**Sol 1:**

[**http://en.wikipedia.org/wiki/Selection\_algorithm#Partition-based\_general\_selection\_algorithm**](http://en.wikipedia.org/wiki/Selection_algorithm#Partition-based_general_selection_algorithm)

Depending on how big the K is, we would split 4GB into 1GB (or less) chunks.   
Then partition-select(k) each. O(n)  
Merge 4 pieces O(n), O(K) space.  
Again, partition-select(k). O(n)  
Multily.O(1)

**Sol2:**

Create Int32 array in size of 134217728, to get a bit for each possible number might be represented as Int32 (this array initiated as default to all 0).  
Read numbers from file one after the other (assuming there is a single number per line)  
For each number read set its corresponding bit (in case it wasn't set already) in the array (calculate appropriate location of corresponding int within the array and set the bit within)  
At all times maintain the maximum number you facing  
after finishing reading the all file go over all int numbers in arrays (starting from the int represents the bit of the largest number backwards), for each int != 0, print corresponding number represented by each set bit, until K bits where observed (or you finished searching the all array).

**Sol3:**

we can do it using minHeap :) lest say file size is s=2^32 byte so number of integer it can have will be N=s/32  
Algorithm  
1.create min-heap of size K  
2. for remaining elements read from file one by one compare it with root of heap if its greatre then root then replace it with root & call heapify  
3. repeat whole algo until EOF  
our heap will contain the k largest elements , return product of all the elemnmts in heap will give us answer  
Time Complexity (O(k)+O(n-k)logk)

## Given a array find the sub array with maximum sum.

Kadane’s algorithm: <http://en.wikipedia.org/wiki/Maximum_subarray_problem>

int GetMaxSum(const vector<int>& array, int\* max\_lb, int\* max\_ub) {

int maxsum = std::numeric\_limits<int>::min();

int sum = 0;

int lb = 0;

\*max\_lb = \*max\_ub = lb;

int len = array.size();

for (int i = 0; i < len; ++i) {

sum += array[i];

if (sum < array[i]) {

sum = array[i];

lb = i;

}

if (maxsum < sum) {

maxsum = sum;

\*max\_lb = lb;

\*max\_ub = i;

}

}

return maxsum;

}

## We have two sorted int arrays: a[] --> size N --> contains N elements; b[] --> size 2N --> contains N elements, and N vacant locations. Write an algorithm of complexity O(n) such that b[] contains elements of a[] and b[] in ascending order.

## Given two sorted arrays a[] and b[]. Merge them and answer should be stored in a[] with minimum complexity

public class MergeArrays {

public static void main(String args[]){

int a[]={1,3,5,0,0,0};

int [] b={2,4,6};

merge (a,3,b,3);

print (a);

}

public static void merge(int[] a,int m,int[] b,int n){

int i=m-1;

int j=n-1;

int k=m+n-1;

while(k>0){

if(a[i]<b[j]){

a[k]=b[j];

k--;

j--;

}

else{

a[k]=a[i];

k--;

i--;

}

}

}

public static void print(int[] a){

for(int i=0;i<a.length;i++){

System.out.print(a[i]+",");

}

}

}

**Coded differently**

void merge(int a[],int b[], int M)

{

int i=M-1,j=M-1,k=M+M-1;

while(i>=0 && j>=0)

{

if(a[i]>b[j])

a[k--]=a[i--];

if(a[i]<=b[j])

{

a[k--]=b[j--];

}

}

}

## Given two arrays a[] = {1,3,2,4} b[] = {4,2,3,1}both will have the same numbers but in different combination. We have to sort them. The condition is that you cannot compare only elements within the same array.

#include <stdio.h>

static int q=0;

void quicksort(int arr[],int b[], int low, int high,int index,int flag);

int main() {

int a[5]={1,4,5,3,2};

int b[5]={3,5,4,2,1};

int i = 0;

quicksort(a,b,0,4,0,0);

printf("\narray1 ");

for(i = 0; i < 5; i++) {

printf(" %d ", a[i]);

}

printf("\narray2 ");

for(i = 0; i < 5; i++) {

printf(" %d ", b[i]);

}

}

void quicksort(int arr[],int arr2[],int low, int high,int z,int flag) {

int i = low;

int j = high;

int y = 0;

int val=arr2[z];

printf("v=%d f=%d q=%d-->",val,flag,q);

int next;

while(i<=j){

while(arr[i] < val ) i++;

while(arr[j] > val ) j--;

if(arr[i]==val || arr[j]==val)

next=i;

if(i <= j) {

y = arr[i];

arr[i] = arr[j];

arr[j] = y;

if(arr[i]==val)

{

j--;

}

if(arr[j]==val)

{ i++; }

}

}

if(q==4)

return;

for(i = 0; i < 5; i++) {

printf(" %d ", arr[i]);

}

printf("\n");

if(!flag)

{

printf("array2\n");

quicksort(arr2,arr,low,high,next,1);

}

if(flag)

{

printf("array 1\n");

quicksort(arr,arr2,low,high,++q,0);

}

}

## Generate the intersection of two sorted sequences

**while**(a[i]&&a[j])

{

**if**(a[i]==a[j])

{

//store result in other array

i++;

j++;

}

**else** **if**(a[i]<a[j])

i++;

**else**

j++;

}

## Find two numbers(a and b) from integer array such that a + b = x, where x is input along with array.

Sol1: Run two loops for each a[i] search for x-a[i] in second loop.

Sol2: Sort the array and for each element a[i], do binary search for x-a[i]

Sol3: Hash the array, do search for x-a[i].

## Given a set of n integers and an integer x. Design an algorithm to check whether k integers add up to x in the given set. The complexity should be O(n^(k-1) \* logn)

Sort the given array(O(nlogn). Consider every possible combinations of k-1 elements in the given array{nC(k-1) = O(n^(k-1). For each combination 'y'=(y1, y2, y3, yk-1), binary search through the given array to find if there is an element not in 'y' that is equal to x - y1 - y2 – y3- yk-1

Total time complexity = O(n^(k-1)\*logn

## Design an algorithm to find all elements that appear more than n/2 times in the list. Then do it for elements that appear more than n/4 times.

Sol1: We can use the median finding for n/2

Sol2: // We don't need an array

public int FindMostFrequentElement(IEnumerable<int> sequence)

{

// Initial value is irrelevant if sequence is non-empty,

// but keeps compiler happy.

int best = 0;

int count = 0;

foreach (int element in sequence)

{

if (count == 0)

{

best = element;

count = 1;

}

else

{

// Vote current choice up or down

count += (best == element) ? 1 : -1;

}

}

return best;

}

For n/4, get the count of all the elements, using counting hash.

## Finding the Median of unsorted array in Linear Time

Let a1, ..., an be n real numbers. We would like to compute their median in linear time. This can be done in linear time by a relatively complicated deterministic algorithm (see any standard book on data-structures). Luckily, it can be also be done in linear time by the following simple algorithm:

FindKMedian( A, K )

Return the number in A which is the K-th in its size.

Pick randomly a number a from A = {a1, ..., an}.

Partition the n numbers into two sets:

S - all the numbers smaller than a

B - all the numbers bigger than a

If |S| = K-1 then a is the required K-median. Return a

If |S| < K-1 then the K-median lies somewhere in B. Call recursively to FindKMedian( B, K - |S| - 1 )

Else, call recursively to FindKMedian( S, K ).

## Given an array of integers for each elemnt in the array find the closest greatest elemnt to the right. Closest means the distance between two elements array indices must be minimum. Problem is to find index of closest greater element to its right.

Sol1: Naïve approach is to run a loop and look for max element in the right subarray. O(n2)

Sol2: O(n)

1. Create a stack of n elements. Push A[0] onto stack.

2. Maintain two variables Min, Max. Initially Min = Max = A[0].

3. for i: 1 to n-1

if A[i] <= Min

Push A[i] onto stack

Min = A[i]

else if A[i] > Max

'i' will be the closest greatest element to the right for all

the elements in the stack

Pop all the elements in the stack and update the result array

else (if A[i] > Min & A[i] <= Max)

pop all elements < A[i]

'i' will be the closest greatest element to the right for all

the popped elements

Update Min = topmost element on the stack

Space: O(n)

Time : O(n)

std::vector<**int**> input; //contains the input array

std::map<**int**,**int**> outputmap; //contains the output as a map

std::stack<**int**> current; // a stack to maintain pairs that have not been found yet

**for** ( size\_t i = 0; i < input.size() ; ++i )

{

**while**( current.size() > 0 && current.top() < input[i] )

{

**int** topElement = current.top();

outputmap[topElement] = input[i];

current.pop();

}

current.push( input[i] );

}

## Given an array A[] and a integer num. Find four no.s in the array whose sum is equal to given num. Brute force :- O(n^4) I solved it in O(n^3)

O(n^4) space O(1)

O(n^3) space

**Sol 1**

Hashtable sums will store all possible sums of two different elements. For each sum S it returns pair of indexes i and j such that a[i] + a[j] == S and i != j. But initially it's empty, we'll populate it on the way.

for (int i = 0; i < n; ++i) {

// 'sums' hastable holds all possible sums a[k] + a[l]

// where k and l are both less than i

for (int j = i + 1; j < n; ++j) {

int current = a[i] + a[j];

int rest = X - current;

// Now we need to find if there're different numbers k and l

// such that a[k] + a[l] == rest and k < i and l < i

// but we have 'sums' hashtable prepared for that

if (sums[rest] != null) {

// found it

}

}

// now let's put in 'sums' hashtable all possible sums

// a[i] + a[k] where k < i

for (int k = 0; k < i; ++k) {

sums[a[i] + a[k]] = pair(i, k);

}

}

**Sol 2**

the method:

1 sort the array

2 give four index a1 a2 a3 a4, set a1 = arr[0] a2 = arr[1] a3 = arr[n-2] a4=arr[n-1]

3 if (a1+a2+a3+a4) > num there are two ways to change them

(1) a3 move to left about one step

(2) a4 move to left about one step if a4 equal with a3

a4 move to left continue and let a4 = max(a4,a3) a3=min(a4,a3)

4 if (a1+a2+a3+a4) == num then find out the result

5 if (a1+a2+a3+a4) < num there also two ways to change them

(1) a1 move right about one step ,if (a1 == a3) a1 move to right continue.a1 = min(a1,a2) a2=max(a1,a2)

(2) a2 move to right about one step

int findFourSubNum(int \*arr,int nLength, int& one, int &two, int &three, int &four, int sum)

{

int cn = arr[one]+arr[two]+arr[three]+arr[four];

if (sum == cn)

return 0;

int cone = one;

int ctwo = two;

int cthree = three;

int cfour = four;

if (cn > sum)

{

cfour--;

if (cfour == cthree)

{

cfour = cthree;

cthree--;

if (cthree <= ctwo)

return -1;

}

if (0 == findFourSubNum(arr,nLength,cone,ctwo,cthree,cfour,sum))

{

equal();

return 0;

}

cthree--;

if (cthree <= ctwo)

return -1;

if (0 == findFourSubNum(arr,nLength,cone,ctwo,cthree,cfour,sum))

{

equal();

return 0;

}

return -1;

}

else

{

cone++;

if (cone == ctwo)

{

cone = ctwo;

ctwo++;

if (ctwo >= cthree)

return -1;

}

if (0 == findFourSubNum(arr,nLength,cone,ctwo,cthree,cfour,sum))

{

equal();

return 0;

}

ctwo++;

if (ctwo >= cthree)

return -1;

if (0 == findFourSubNum(arr,nLength,cone,ctwo,cthree,cfour,sum))

{

equal();

return 0;

}

return -1;

}

}

int getFourSubNum(int \*arr, int nLength, int sum)

{

if (NULL == arr)

return 0;

quickSort(arr,0,nLength-1);

int one = 0,two = 1,three = nLength-2,four = nLength-1;

int re = findFourSubNum(arr,nLength,one,two,three,four,sum);

if (0 == re)

{

printf("%d+%d+%d+%d = %d\n",arr[one],arr[two],arr[three],arr[four],sum);

}

return 0;

}

## Given an array which consists of elements in the following form : ->All the adjacent elements differ only by value -1 or +1. ->You are given an element. You need to search for its index.

int findIndex(int[] array, int n, int value){

int head=0;

int end=n-1;

while(head<end){

if(array[head]==value){

return head;

}

else{

head=head+Math.abs(value-array[head]);

}

if(array[end]==value){

return end;

}

else{

end=end-Math.abs(value-array[end]);

}

}

return -1;

}

## Given a doubly linked list with just 3 numbers 0,1,2 . Sort it

**Dutch national flag problem**

struct node\* sort012(struct node \*head)

{

int count=0,itercount=0;

struct node \*lo = head;

struct node \*hi = last(head,&count);

struct node \*mid = head;

while(itercount++<count)

{

switch(mid->data)

{

case 0:

swap(&lo->data,&mid->data);

lo=lo->next;

mid=mid->next;

break;

case 1:

mid=mid->next;

break;

case 2:

swap(&mid->data, &hi->data);

hi=hi->prev;

break;

}

}

}

## Dutch National flag problem

Partition the array into three parts:

P1 <low;

Low<=P2 < high;

P3 >=high

void threeWayPartition(int data[], int size, int low, int high) {

int p = -1;

int q = size;

for (int i = 0; i < q;) {

if (data[i] < low) {

swap(data[i], data[++p]);

++i;

} else if (data[i] >= high) {

swap(data[i], data[--q]); // when swapping high side i doesn't move

} else {

++i;

}

}

}

## How to find the max product of three numbers out of all elements of an array of integers.

Let Mn be nth greatest +ve number. M1 M2 M3 are three max positive numbers (consider 0 as positive).

Similarly let m1 m2 m3 be 3 max -ve numbers (max by actual value and not absolute value)

and L1 L2 L3 be the 3 smallest +ve number

and l1 l2 l3 be the 3 smallest -ve number

Calculate M1 to M3 and similarly L1..3 m1..3 l1..3 (This is linear time)

There are 4 possibilities to get max product

1. ++- ie 2 +ve and 1 -ve number.

Max product is L1\*L2\*m1 = p1

2. --+

Max product is l1\*l2\*M1 = p2

3. ---

Max product is m1\*m2\*m3 = p3

4. +++

Max product is M1\*M2\*M3 = p4

Calculate p1 p2 p3 p4 and check for maximum among them and then return appropriate integers.

This will run in linear time.

## You are given some denominations of coins in an array (int denom[])and infinite supply of all of them. Given an amount (int amount), find the minimum number of coins required to get the exact amount. What is the method called?

0/1Knapsack problem

Given a list of N coins, their values (**V1**, **V2**, ... , **VN**), and the total sum **S**. Find the minimum number of coins the sum of which is **S** (we can use as many coins of one type as we want), or report that it's not possible to select coins in such a way that they sum up to **S**.   
  
Now let's start constructing a DP solution:   
  
First of all we need to find a state for which an optimal solution is found and with the help of which we can find the optimal solution for the next state.   
  
***What does a "state" stand for?***   
  
It's a way to describe a situation, a sub-solution for the problem. For example a state would be the solution for sum **i**, where **i≤S**. A smaller state than state **i** would be the solution for any sum **j**, where **j<i**. For finding a **state i**, we find all need to first smaller states **j (j<i)**. Having found the minimum number of coins which sum up to **i**, we can easily find the next state - the solution for **i+1**.   
  
***How can we find it?***   
  
It is simple - for each coin **j, Vj≤i**, look at the minimum number of coins found for the **i-Vj**sum (we have already found it previously). Let this number be **m**. If **m+1** is less than the minimum number of coins already found for current sum **i**, then we write the new result for it.   
  
For a better understanding let's take this example:  
Given coins with values 1, 3, and 5.  
And the sum **S** is set to be 11.   
  
First of all we mark that for state 0 (sum 0) we have found a solution with a minimum number of 0 coins. We then go to sum 1. First, we mark that we haven't yet found a solution for this one (a value of Infinity would be fine). Then we see that only coin 1 is less than or equal to the current sum. Analyzing it, we see that for sum 1-**V1**= 0 we have a solution with 0 coins. Because we add one coin to this solution, we'll have a solution with 1 coin for sum 1. It's the only solution yet found for this sum. We write (save) it. Then we proceed to the next state - **sum 2**. We again see that the only coin which is less or equal to this sum is the first coin, having a value of 1. The optimal solution found for sum (2-1) = 1 is coin 1. This coin 1 plus the first coin will sum up to 2, and thus make a sum of 2 with the help of only 2 coins. This is the best and only solution for sum 2. Now we proceed to sum 3. We now have 2 coins which are to be analyzed - first and second one, having values of 1 and 3. Let's see the first one. There exists a solution for sum 2 (3 - 1) and therefore we can construct from it a solution for sum 3 by adding the first coin to it. Because the best solution for sum 2 that we found has 2 coins, the new solution for sum 3 will have 3 coins. Now let's take the second coin with value equal to 3. The sum for which this coin needs to be added to make 3 , is 0. We know that sum 0 is made up of 0 coins. Thus we can make a sum of 3 with only one coin - 3. We see that it's better than the previous found solution for sum 3 , which was composed of 3 coins. We update it and mark it as having only 1 coin. The same we do for sum 4, and get a solution of 2 coins - 1+3. And so on.   
  
**Pseudocode:**

Set Min[i] equal to Infinity for all of i

Min[0]=0

For i = 1 to S

For j = 0 to N - 1

If (Vj<=i AND Min[i-Vj]+1<Min[i])

Then Min[i]=Min[i-Vj]+1

Output Min[S]

Here are the solutions found for all sums:

|  |  |  |
| --- | --- | --- |
| **Sum** | **Min. nr. of coins** | **Coin value added to a smaller sum to obtain this sum (it is displayed in brackets)** |
| 0 | 0 | - |
| 1 | 1 | 1 (0) |
| 2 | 2 | 1 (1) |
| 3 | 1 | 3 (0) |
| 4 | 2 | 1 (3) |
| 5 | 1 | 5 (0) |
| 6 | 2 | 3 (3) |
| 7 | 3 | 1 (6) |
| 8 | 2 | 3 (5) |
| 9 | 3 | 1 (8) |
| 10 | 2 | 5 (5) |
| 11 | 3 | 1 (10) |

As a result we have found a solution of 3 coins which sum up to 11. 

## Find the maximum sum subsequence in an array with positive nos such that no two nos are adjacent. Should explain code, write the code with base conditions properly, and give all possible test cases.

#include <stdio.h>

#define MAX 9

int main(){

int a[MAX]={10,3,9,11,4,7,6,2,8};

int part\_sum[MAX]; //holds all the part sums part\_sum[i] holds max sum of a[i....n]

int taken[MAX]={0}; //denotes whether a particular element is taken in the part sum or not in the max sum of a[i....n]

int temp1,temp2,i;

for(i=MAX-1;i>=0;i--){

if(i==MAX-1){

part\_sum[i]=a[i];

taken[i]=1;

}

else if(i==MAX-2){

if(a[i]>a[i+1]){

part\_sum[i]=a[i];

taken[i]=1;

}else{

part\_sum[i]=a[i+1];

taken[i]=0;

}

}

else{

temp1=a[i]+part\_sum[i+2];

temp2=part\_sum[i+1];

if(temp1>temp2){

part\_sum[i]=temp1;

taken[i]=1;

}

else{

part\_sum[i]=temp2;

taken[i]=0;

}

}

}

printf("Maximum sum possible is: %d\n",part\_sum[0]);

printf("\nThe maximum subsequence is:\n");

i=0;

while(i<MAX){

if(taken[i]==0)i=i+1;

else{

printf("%d,",a[i]);

i=i+2;

}

}

return 0;

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Array | 9 | 4 | 7 | 15 | 5 |
| Part\_sum | 24 | 19 | 15 | 15 | 5 |
| taken | 1 | 1 | 0 | 1 | 1 |

24

9, 15

## Given an array all of whose elements are positive numbers. Find the maximum sum of a subsequence with the constraint that no two numbers in the sequence should be adjacent in the array. So 3 2 7 10 should return 13 (sum of 3 and 10) or 3 2 5 10 7 should return 15 (sum of 3, 5 and 7)

This problem can be solved by dynamic programming.

Let's say we have an array of integers:

i[1], i[2], i[3], ..., i[n], i[n+1], i[n+2]

We partition the array into two parts: the first part containing first n integers, and the second part is the last two integers:

{i[1], i[2], i[3], ..., i[n]}, {i[n+1], i[n+2]}

Let's denote M\_SUM(n) as the max sum of the first n integers per your requirement.

There will be two cases:

1. if i[n] is not counted into M\_SUM(n), then M\_SUM(n+2) = M\_SUM(n) + MAX(i[n+1], i[n+2])
2. if i[n] is counted into M\_SUM(n), then M\_SUM(n+2) = M\_SUM(n) + i[n+2]

then, M\_SUM(n+2), the value we are seeking for, will be the larger value of the above two.

Then we can have a very naive pseudocode as below:

function M\_SUM(n)  
   return MAX(M\_SUM(n, true), M\_SUM(n, false))  
  
function M\_SUM(n, flag)  
   if n == 0 then return 0  
   else if n == 1  
      return flag ? i[0] : 0  
   } else {  
      if flag then  
         return MAX(  
                M\_SUM(n-2, true) + i[n-1],   
                M\_SUM(n-2, false) + MAX(i[n-1],i[n-2]))  
      else  
         return MAX(M\_SUM(n-2, false) + i[n-2], M\_SUM(n-2, true))  
   }

"flag" means "allow using the last integer"

This algorithm has an exponential time complexity.

Dynamical programming techniques can be employed to eliminate the unnecessary recomputation of M\_SUM.

Storing each M\_SUM(n, flag) into a n\*2 matrix. In the recursion part, if such a value does not present in the matrix, compute it. Otherwise, just fetch the value from the matrix. This will reduce the time complexity into linear.

The algorithm will have O(n) time complexity, and O(n) space complexity.

## Find the second largest element in an array with minimum no of comparisons and give the minimum no of comparisons needed on an array of size N to do the same.

You can use a modified quick sort kind of algo.. and do a quick select.

## Given a string, find the first un-repeated character in it? Give some test cases

O(n2)

char FirstNonRepeatedChar(char \* psz)

{

for (int ii = 0; psz[ii] != 0; ++ii)

{

for (int jj = ii+1; ; ++jj)

{

// if we hit the end of string, then we found a non-repeat character.

//

if (psz[jj] == 0)

return psz[ii]; // this character doesn't repeat

// if we found a repeat character, we can stop looking.

//

if (psz[ii] == psz[jj])

break;

}

}

return 0; // there were no non-repeating characters.

}

O(n)

def first\_non\_repeated\_character(string):

chars = [] //Non repeating chars

repeated = [] //Repeating chars

for character in string: //For every character in string

if character in chars: //if character is in non-r-chars hash.

chars.remove(character) //remove the character from n-r-c hash

repeated.append(character) //and append it to repeated hash

else:

if not character in repeated: //else if character is not in repeated hash

chars.append(character) //add the character to n-r-c hash.

if len(chars):

return chars[0]

else:

return False

Code in java using map

public class FirstNonRepeatingChar

{

public static void main(String[] args)

{

System.out.println("[FirstNonRepeatingChar] in main()");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the string:");

try {

String str= br.readLine();

Map<Character,Integer> map = new HashMap();

for( int i=0;i<str.length();i++ )

{

Character ch =str.charAt(i);

if(map.get(ch)==null)

map.put(ch, 1);

else

map.put(ch, map.get(ch)+1);

}

int i=0;

for( ;i<str.length();i++ )

{

Character ch =str.charAt(i);

if(map.get(ch)==1)

{

System.out.println("First Non-repeated character is: "+ch);

break;

}

}

if(i==str.length())

{

System.out.println("No Non-repeated character ");

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

## Given an array which is sorted in ascending order upto kth element and then in descending order. Find the maximum element.

1. pick the middle and check

a. A[mid-1]<A[mid]<A[mid+1] if true, this means the transition point is between mid and the right  
b. A[mid-1]>A[mid]>A[mid+1] if true, this means the transition point is between mid and the left

also check if

c. A[mid-1]<A[mid] && A[mid]> A[mid+1] this means the mid is the transition point.

recursively pick the side that contains the transition point.

public static int findMax(int[] A, int left, int right){

if(right-left==1){

if(A[left]>A[right])

return left;

else

return right;

}

int mid = (left+right)/2;

if(A[mid]>A[mid-1] && A[mid]>=A[mid+1]){

return mid;

}

if(A[mid]>A[mid-1] && A[mid]<=A[mid+1]){

return findMax(A, mid+1, right);

}

if(A[mid]<A[mid-1] && A[mid]>=A[mid+1]){

return findMax(A, left, mid-1);

}

return -1;

}

## Sort an array of n positive integers containing n/2 sorted integers in first and second-half?

in O(n) time complexity ..

and space complexity should be constant

class InplaceMerger

{

public:

static void MergeInplace(std::vector<int>& arr)

{

auto current1 = arr.begin();

auto current2 = arr.begin() + arr.size()/2;

auto end = arr.end();

while(current1 != end && current2 != end)

{

if(\*current1 > \*current2)

{

std::swap(\*current1, \*current2);

}

if(current1 + 1 == current2) current2++;

else current1++;

}

}

};

current1: 2 current2: 3 [8](2,4,7,9,3,5,10,12)

current1: 4 current2: 3 [8](2,4,7,9,3,5,10,12)

current1: 7 current2: 4 [8](2,3,7,9,4,5,10,12)

current1: 9 current2: 7 [8](2,3,4,9,7,5,10,12)

current1: 7 current2: 5 [8](2,3,4,7,9,5,10,12)

current1: 9 current2: 7 [8](2,3,4,5,9,7,10,12)

current1: 7 current2: 10 [8](2,3,4,5,7,9,10,12)

current1: 9 current2: 10 [8](2,3,4,5,7,9,10,12)

current1: 9 current2: 12 [8](2,3,4,5,7,9,10,12)

current1: 10 current2: 12 [8](2,3,4,5,7,9,10,12)

current1: 10 current2: -842150451 [8](2,3,4,5,7,9,10,12)

## How to find duplicate element (only one element is repeated) from an array of unsorted positive integers..

time complexity .. O(n)

space .. o(1).

sum the numbers

if n is known (n-1)(n-2)/2

Ans = sum - (n-1)elements sum

## Given a 2d array sorted in increasing order from left to right and top to bottom, what is the best way to search for a target number?

bool numberSearch(int[][] arr, int value, int minX, int maxX, int minY, int maxY)

{

if (minX == maxX and minY == maxY and arr[minX,minY] != value)

return false;

if (arr[minX,minY] > value)

return false; // Early exits if the value can't be in

if (arr[maxX,maxY] < value)

return false; // this subrange at all.

int nextX = (minX + maxX) / 2;

int nextY = (minY + maxY) / 2;

if (arr[nextX,nextY] == value)

{

print nextX,nextY;

return true;

}

else if (arr[nextX,nextY] < value)

{

if (numberSearch(arr, value, minX, maxX, nextY + 1, maxY))

return true;

return numberSearch(arr, value, nextX + 1, maxX, minY, nextY);

}

else

{

if (numberSearch(arr, value, minX, nextX - 1, minY, maxY))

return true;

return numberSearch(arr, value, nextX, maxX, minY, nextY);

}

}

Better Sol2:

1) Start with top right element  
2) Loop: compare this element e with x  
….i) if they are equal then return its position  
…ii) e < x then move it to down (if out of bound of matrix then break return false)  
..iii) e > x then move it to left (if out of bound of matrix then break return false)  
3) repeat the i), ii) and iii) till you find element or returned false

**Implementation:**

|  |
| --- |
| #include<stdio.h>  /\* Searches the element x in mat[][]. If the element is found,  then prints its position and returns true, otherwise prints  "not found" and returns false \*/  int search(int mat[4][4], int n, int x)  {  int i = 0, j = n-1; //set indexes for top right element  while ( i < n && j >= 0 )  {  if ( mat[i][j] == x )  {  printf("\n Found at %d, %d", i, j);  return 1;  }  if ( mat[i][j] > x )  j--;  else // if mat[i][j] < x  i++;  }  printf("\n Element not found");  return 0; // if ( i==n || j== -1 )  }  // driver program to test above function  int main()  {  int mat[4][4] = { {10, 20, 30, 40},  {15, 25, 35, 45},  {27, 29, 37, 48},  {32, 33, 39, 50},  };  search(mat, 4, 29);  getchar();  return 0;  } |

Time Complexity: O(n)

The above approach will also work for m x n matrix (not only for n x n). Complexity would be O(m + n).

## We have a N X N matrix whose rows and columns are in sorted order. How effeciently can we find the median of those N^2 keys ?

<http://www.careercup.com/question?id=10506663>

## Find 2 numbers that add to given sum in an array.

Steps:  
1) Sort the array in nlog(n) time.  
2) If sum of first and last element is == given number. Bingo !!  
3) Else if sum is greater, do sum of first and second last element and check.  
4) Else if sum is less, do sum of second and last element and check.  
Repeat step 3 and 4 until you find the given sum OR the given number lies between the sum of previous computation and present computation, in that case sum does not exists.

public static boolean findSum2(int[] a, int sum) {

if (a.length == 0) {

return false;

}

Arrays.sort(a);

int i = 0;

int j = a.length - 1;

while (i < j) {

int tmp = a[i] + a[j];

if (tmp == sum) {

System.out.println(a[i] + "+" + a[j] + "=" + sum);

return true;

} else if (tmp > sum) {

j--;

} else {

i++;

}

}

return false;

}

## You are given a dictionary of all valid words. You have the following 3 operations permitted on a word: delete a character, insert a character, replace a character. Now given two words - word1 and word2 - find the minimum number of steps required to convert word1 to word2. (one operation counts as 1 step.)

Using Dynamic programming and doing sequence alignment.

## Merge two sorted integer arrays in place where the second one has space for the elements in the first.

A[n]

B[m+n]

One way is to keep shifting the second array’s elements whenever we insert first array’s elements.

O(mn)

The other way is to put the elements of A in B and sort B.

## Given a set of strings, check whether it is possible to chain all of them. Two strings can be chained iff s1[n] == s2[0] || s2[0] == s1[n]

notice that only the first and last characters matter. so lets say you have strings "AB", "CD", "BC, "BD", "DA"  
  
so you have a graph which nodes are the characters A, B, C, D. now, the strings are directed edges from A to B, from C to D, from B to C, from B to D and from D to A. what you want is a path through every edge. that is exactly a chaining of the strings. it can be done on linear time.

Euler walk:

## Given huge file having list of numbers find the largest possible sum of any k numbers

Maintain a array of size k. a[k], This array will remain sorted, and we will parse through the file and keep on adding numbers to the array. If the number just read is greater than the smallest number in array we will replace the minimum number and insert new number in the array.

Heap can be used here. Maintain a max heap of K numbers.

## Given an array of strings, we have to write a function called reverseStringSort which have to reverse the given strings, sort it and again reverse it finally before displaying it.

We might not need to reverse the strings twice; we can use the comparator that compares the string backwards.

public static string[] SortStringArray(string[] strarr)

{

string temp;

for (int i = 0; i < strarr.Length; i++)

{

for (int k = 0; k < strarr.Length; k++)

{

if(strarr[i][strarr[i].Length-1] < strarr[k][strarr[k].Length-1])

{

temp = strarr[i];

strarr[i] = strarr[k];

strarr[k] = temp;

}

}

}

return strarr;

}

## Write C code to implement the strstr() (search for a substring) function.

**Sol1 :**

void BruteForce(char \*x /\* pattern \*/,

int m /\* length of the pattern \*/,

char \*y /\* actual string being searched \*/,

int n /\* length of this string \*/)

{

int i, j;

printf("\nstring : [%s]"

"\nlength : [%d]"

"\npattern : [%s]"

"\nlength : [%d]\n\n", y,n,x,m);

/\* Searching \*/

for (j = 0; j <= (n - m); ++j)

{

for (i = 0; i < m && x[i] == y[i + j]; ++i);

if (i >= m) {printf("\nMatch found at\n\n->[%d]\n->[%s]\n",j,y+j);}

}

}

int main()

{

char \*string = "hereroheroero";

char \*pattern = "hero";

BF(pattern,strlen(pattern),string,strlen(string));

printf("\n\n");

return(0);

}

**Sol2:**

The second method is called the Rabin-Karp method.

Instead of checking at each position of the text if the pattern occurs or not, it is better to check first if the contents of the current string "window" looks like the pattern or not. In order to check the resemblance between these two patterns, a hashing function is used. Hashing a string involves computing a numerical value from the value of its characters using a hash function.

The Rabin-Karp method uses the rule that if two strings are equal, their hash values must also be equal. Note that the converse of this statement is not always true, but a good hash function tries to reduce the number of such hash collisions. Rabin-Karp computes hash value of the pattern, and then goes through the string computing hash values of all of its substrings and checking if the pattern's hash value is equal to the substring hash value, and advancing by 1 character every time. If the two hash values are the same, then the algorithm verifies if the two string really are equal, rather than this being a fluke of the hashing scheme. It uses regular string comparison for this final check. Rabin-Karp is an algorithm of choice for multiple pattern search. If we want to find any of a large number, say k, fixed length patterns in a text, a variant Rabin-Karp that uses a hash table to check whether the hash of a given string belongs to a set of hash values of patterns we are looking for. Other algorithms can search for a single pattern in time order O(n), hence they will search for k patterns in time order O(n\*k). The variant Rabin-Karp will still work in time order O(n) in the best and average case because a hash table allows to check whether or not substring hash equals any of the pattern hashes in time order of O(1).

Here is some code (not working though!)

#include

hashing\_function()

{ // A hashing function to compute the hash values of the strings.

....

}void KarpRabinR(char \*x, int m, char \*y, int n)

{ int hx, hy, i, j;

printf("\nstring : [%s]"

"\nlength : [%d]"

"\npattern : [%s]"

"\nlength : [%d]\n\n", y,n,x,m);

/\* Preprocessing phase \*/

Do preprocessing here..

/\* Searching \*/

j = 0;

while (j <= n-m)

{

if (hx == hy && memcmp(x, y + j, m) == 0)

{

// Hashes match and so do the actual strings!

printf("\nMatch found at : [%d]\n",j);

}

hy = hashing\_function(y[j], y[j + m], hy);

++j;

}

}int main()

{ char \*string="hereroheroero";

char \*pattern="hero";

KarpRabin(pattern,strlen(pattern),string,strlen(string));

printf("\n\n");

return(0);

}

Sol 3

Knuth-Morris-Pratt or the Morris-Pratt algorithms

The Knuth-Morris-Pratt or the Morris-Pratt algorithms are extensions of the basic Brute Force algorithm. They use precomputed data to skip forward not by 1 character, but by as many as possible for the search to succeed.

Here is some code

void preComputeData(char \*x, int m, int Next[])

{

int i, j;

i = 0;

j = Next[0] = -1;

while (i < m)

{

while (j > -1 && x[i] != x[j])

j = Next[j];

Next[++i] = ++j;

}

}

void MorrisPrat(char \*x, int m, char \*y, int n)

{

int i, j, Next[1000];

/\* Preprocessing \*/

preComputeData(x, m, Next);

/\* Searching \*/

i = j = 0;

while (j < n)

{

while (i > -1 && x[i] != y[j])

i = Next[i];

i++;

j++;

if (i >= m)

{

printf("\nMatch found at : [%d]\n",j - i);

i = Next[i];

}

}

}

int main()

{

char \*string="hereroheroero";

char \*pattern="hero";

MorrisPrat(pattern,strlen(pattern),string,strlen(string));

printf("\n\n");

return(0);

}

Sol4 :Suffix tree

## Given a list of words, identify words which are anagrams of each other, and print them out as sets of anagrams.

Sort the letters in the words, put the sorted word in a hash.

If a match is found add then create a new anagram list and add these two to them. OR we can use the hash bucket to put the words there.

## Write a function to check if the two strings given are anagrams are not

bool IsAnagram(char\* s1, char\* s2) {

char hash[256] = { 0 };

if (strlen(s1) != strlen(s2)) return false;

for(int i=0;i<strlen(s1);i++)

hash[s1[i]]++;

for(int i=0;i<strlen(s2);i++) {

if(s1[i] == s2[i] ||

--hash[s2[i]] < 0) return false;

}

return true;

}

## Median of two sorted arrays

**Method 1 (Simply count while Merging)**  
Use merge procedure of merge sort. Keep track of count while comparing elements of two arrays. If count becomes n(For 2n elements), we have reached the median. Take the average of the elements at indexes n-1 and n in the merged array.

/\* This function returns median of ar1[] and ar2[].

Assumptions in this function:

Both ar1[] and ar2[] are sorted arrays

Both have n elements \*/

**int** getMedian(**int** ar1[], **int** ar2[], **int** n)

{

**int** i = 0; /\* Current index of i/p array ar1[] \*/

**int** j = 0; /\* Current index of i/p array ar2[] \*/

**int** count;

**int** m1 = -1, m2 = -1;

/\* Since there are 2n elements, median will be average

of elements at index n-1 and n in the array obtained after

merging ar1 and ar2 \*/

**for** (count = 0; count <= n; count++)

{

/\*Below is to handle case where all elements of ar1[] are

smaller than smallest(or first) element of ar2[]\*/

**if** (i == n)

{

m1 = m2;

m2 = ar2[0];

**break**;

}

/\*Below is to handle case where all elements of ar2[] are

smaller than smallest(or first) element of ar1[]\*/

**else** **if** (j == n)

{

m1 = m2;

m2 = ar1[0];

**break**;

}

**if** (ar1[i] < ar2[j])

{

m1 = m2; /\* Store the prev median \*/

m2 = ar1[i];

i++;

}

**else**

{

m1 = m2; /\* Store the prev median \*/

m2 = ar2[j];

j++;

}

}

**return** (m1 + m2)/2;

}

**Method 2 (By comparing the medians of two arrays)**  
This method works by first getting medians of the two sorted arrays and then comparing them.

Let ar1 and ar2 be the input arrays.

Algorithm:

1) Calculate the medians m1 and m2 of the input arrays ar1[]

and ar2[] respectively.

2) If m1 and m2 both are equal then we are done.

return m1 (or m2)

3) If m1 is greater than m2, then median is present in one

of the below two subarrays.

a) From first element of ar1 to m1 (ar1[0...|\_n/2\_|])

b) From m2 to last element of ar2 (ar2[|\_n/2\_|...n-1])

4) If m2 is greater than m1, then median is present in one

of the below two subarrays.

a) From m1 to last element of ar1 (ar1[|\_n/2\_|...n-1])

b) From first element of ar2 to m2 (ar2[0...|\_n/2\_|])

5) Repeat the above process until size of both the subarrays

becomes 2.

6) If size of the two arrays is 2 then use below formula to get

the median.

Median = (max(ar1[0], ar2[0]) + min(ar1[1], ar2[1]))/2

|  |
| --- |
| #include<stdio.h>    int max(int, int); /\* to get maximum of two integers \*/  int min(int, int); /\* to get minimum of two integeres \*/  int median(int [], int); /\* to get median of a sorted array \*/    /\* This function returns median of ar1[] and ar2[].     Assumptions in this function:     Both ar1[] and ar2[] are sorted arrays     Both have n elements \*/  int getMedian(int ar1[], int ar2[], int n)  {      int m1; /\* For median of ar1 \*/      int m2; /\* For median of ar2 \*/        /\* return -1  for invalid input \*/      if (n <= 0)          return -1;        if (n == 1)          return (ar1[0] + ar2[0])/2;        if (n == 2)          return (max(ar1[0], ar2[0]) + min(ar1[1], ar2[1])) / 2;        m1 = median(ar1, n); /\* get the median of the first array \*/      m2 = median(ar2, n); /\* get the median of the second array \*/        /\* If medians are equal then return either m1 or m2 \*/      if (m1 == m2)          return m1;         /\* if m1 < m2 then median must exist in ar1[m1....] and ar2[....m2] \*/      if (m1 < m2)      {          if (n % 2 == 0)              return getMedian(ar1 + n/2 - 1, ar2, n - n/2 +1);          else              return getMedian(ar1 + n/2, ar2, n - n/2);      }        /\* if m1 > m2 then median must exist in ar1[....m1] and ar2[m2...] \*/      else      {          if (n % 2 == 0)              return getMedian(ar2 + n/2 - 1, ar1, n - n/2 + 1);          else              return getMedian(ar2 + n/2, ar1, n - n/2);      }  }    /\* Function to get median of a sorted array \*/  int median(int arr[], int n)  {      if (n%2 == 0)          return (arr[n/2] + arr[n/2-1])/2;      else          return arr[n/2];  }    /\* Driver program to test above function \*/  int main()  {      int ar1[] = {1, 2, 3, 6};      int ar2[] = {4, 6, 8, 10};      int n1 = sizeof(ar1)/sizeof(ar1[0]);      int n2 = sizeof(ar2)/sizeof(ar2[0]);      if (n1 == n2)        printf("Median is %d", getMedian(ar1, ar2, n1));      else       printf("Doesn't work for arrays of unequal size");        getchar();      return 0;  }    /\* Utility functions \*/  int max(int x, int y)  {      return x > y? x : y;  }    int min(int x, int y)  {      return x > y? y : x;  } |

Time Complexity: O(logn)  
Algorithmic Paradigm: Divide and Conquer

**Method 3 (By doing binary search for the median):**  
The basic idea is that if you are given two arrays ar1[] and ar2[] and know the length of each, you can check whether an element ar1[i] is the median in constant time. Suppose that the median is ar1[i]. Since the array is sorted, it is greater than exactly i values in array ar1[]. Then if it is the median, it is also greater than exactly j = n – i – 1 elements in ar2[].  
It requires constant time to check if ar2[j] <= ar1[i] <= ar2[j + 1]. If ar1[i] is not the median, then depending on whether ar1[i] is greater or less than ar2[j] and ar2[j + 1], you know that ar1[i] is either greater than or less than the median. Thus you can binary search for median in O(lg n) worst-case time.

For two arrays ar1 and ar2, first do binary search in ar1[]. If you reach at the end (left or right) of the first array and don't find median, start searching in the second array ar2[].

1) Get the middle element of ar1[] using array indexes left and right.

Let index of the middle element be i.

2) Calculate the corresponding index j of ar2[]

j = n – i – 1

3) If ar1[i] >= ar2[j] and ar1[i] <= ar2[j+1] then ar1[i] and ar2[j]

are the middle elements.

return average of ar2[j] and ar1[i]

4) If ar1[i] is greater than both ar2[j] and ar2[j+1] then

do binary search in left half (i.e., arr[left ... i-1])

5) If ar1[i] is smaller than both ar2[j] and ar2[j+1] then

do binary search in right half (i.e., arr[i+1....right])

6) If you reach at any corner of ar1[] then do binary search in ar2[]

Example:

ar1[] = {1, 5, 7, 10, 13}

ar2[] = {11, 15, 23, 30, 45}

Middle element of ar1[] is 7. Let us compare 7 with 23 and 30, since 7 smaller than both 23 and 30, move to right in ar1[]. Do binary search in {10, 13}, this step will pick 10. Now compare 10 with 15 and 23. Since 10 is smaller than both 15 and 23, again move to right. Only 13 is there in right side now. Since 13 is greater than 11 and smaller than 15, terminate here. We have got the median as 12 (average of 11 and 13)

Implementation:

|  |
| --- |
| #include<stdio.h>    int getMedianRec(int ar1[], int ar2[], int left, int right, int n);    /\* This function returns median of ar1[] and ar2[].     Assumptions in this function:     Both ar1[] and ar2[] are sorted arrays     Both have n elements \*/  int getMedian(int ar1[], int ar2[], int n)  {      return getMedianRec(ar1, ar2, 0, n-1, n);  }    /\* A recursive function to get the median of ar1[] and ar2[]     using binary search \*/  int getMedianRec(int ar1[], int ar2[], int left, int right, int n)  {      int i, j;        /\* We have reached at the end (left or right) of ar1[] \*/      if(left > right)          return getMedianRec(ar2, ar1, 0, n-1, n);        i = (left + right)/2;      j = n - i - 1;  /\* Index of ar2[] \*/        /\* Recursion terminates here.\*/      if (ar1[i] > ar2[j] && (j == n-1 || ar1[i] <= ar2[j+1]))      {          /\*ar1[i] is decided as median 2, now select the median 1             (element just before ar1[i] in merged array) to get the             average of both\*/          if (ar2[j] > ar1[i-1] || i == 0)              return (ar1[i] + ar2[j])/2;          else              return (ar1[i] + ar1[i-1])/2;      }        /\*Search in left half of ar1[]\*/      else if (ar1[i] > ar2[j] && j != n-1 && ar1[i] > ar2[j+1])          return getMedianRec(ar1, ar2, left, i-1, n);        /\*Search in right half of ar1[]\*/      else /\* ar1[i] is smaller than both ar2[j] and ar2[j+1]\*/          return getMedianRec(ar1, ar2, i+1, right, n);  } |

The above solutions can be optimized for the cases when all elements of one array are smaller than all elements of other array. For example, in method 3, we can change the getMedian() function to following so that these cases can be handled in O(1) time.

|  |
| --- |
| /\* This function returns median of ar1[] and ar2[].  Assumptions in this function:  Both ar1[] and ar2[] are sorted arrays  Both have n elements \*/  int getMedian(int ar1[], int ar2[], int n)  {  // If all elements of array 1 are smaller then  // median is average of last element of ar1 and  // first element of ar2  if (ar1[n-1] < ar2[0])  return (ar1[n-1]+ar2[0])/2;  // If all elements of array 1 are smaller then  // median is average of first element of ar1 and  // last element of ar2  if (ar2[n-1] < ar1[0])  return (ar2[n-1]+ar1[0])/2;  return getMedianRec(ar1, ar2, 0, n-1, n);  } |

## Implement a func node \*(char \*word){} which returns a link list of words that are anagrams with the input word..if no anagrams found return NULL and add that word to the link list

Count the number of instances of each letter in the word -- call this the word's signature. Find all words with the same signature.

## Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1.

Example 1

The matrix

1 0

0 0

should be changed to following

1 1

1 0

Example 2

The matrix

1 0 0 1

0 0 1 0

0 0 0 0

should be changed to following

1 1 1 1

1 1 1 1

1 0 1 1

**Method 1 (Use two temporary arrays)**  
1) Create two temporary arrays row[M] and col[N]. Initialize all values of row[] and col[] as 0.  
2) Traverse the input matrix mat[M][N]. If you see an entry mat[i][j] as true, then mark row[i] and col[j] as true.  
3) Traverse the input matrix mat[M][N] again. For each entry mat[i][j], check the values of row[i] and col[j]. If any of the two values (row[i] or col[j]) is true, then mark mat[i][j] as true.

#include <stdio.h>

#define R 3

#define C 4

void modifyMatrix(bool mat[R][C])

{

bool row[R];

bool col[C];

int i, j;

/\* Initialize all values of row[] as 0 \*/

for (i = 0; i < R; i++)

{

row[i] = 0;

}

/\* Initialize all values of col[] as 0 \*/

for (i = 0; i < C; i++)

{

col[i] = 0;

}

/\* Store the rows and columns to be marked as 1 in row[] and col[]

arrays respectively \*/

for (i = 0; i < R; i++)

{

for (j = 0; j < C; j++)

{

if (mat[i][j] == 1)

{

row[i] = 1;

col[j] = 1;

}

}

}

/\* Modify the input matrix mat[] using the above constructed row[] and

col[] arrays \*/

for (i = 0; i < R; i++)

{

for (j = 0; j < C; j++)

{

if ( row[i] == 1 || col[j] == 1 )

{

mat[i][j] = 1;

}

}

}

}

/\* A utility function to print a 2D matrix \*/

void printMatrix(bool mat[R][C])

{

int i, j;

for (i = 0; i < R; i++)

{

for (j = 0; j < C; j++)

{

printf("%d ", mat[i][j]);

}

printf("\n");

}

}

/\* Driver program to test above functions \*/

int main()

{

bool mat[R][C] = { {1, 0, 0, 1},

{0, 0, 1, 0},

{0, 0, 0, 0},

};

printf("Input Matrix \n");

printMatrix(mat);

modifyMatrix(mat);

printf("Matrix after modification \n");

printMatrix(mat);

return 0;

}

Output:

Input Matrix

1 0 0 1

0 0 1 0

0 0 0 0

Matrix after modification

1 1 1 1

1 1 1 1

1 0 1 1

Time Complexity: O(M\*N)  
Auxiliary Space: O(M + N)

**Method 2 (A Space Optimized Version of Method 1)**  
This method is a space optimized version of above method 1. This method uses the first row and first column of the input matrix in place of the auxiliary arrays row[] and col[] of method 1. So what we do is: first take care of first row and column and store the info about these two in two flag variables rowFlag and colFlag. Once we have this info, we can use first row and first column as auxiliary arrays and apply method 1 for submatrix (matrix excluding first row and first column) of size (M-1)\*(N-1).

1) Scan the first row and set a variable rowFlag to indicate whether we need to set all 1s in first row or not.  
2) Scan the first column and set a variable colFlag to indicate whether we need to set all 1s in first column or not.  
3) Use first row and first column as the auxiliary arrays row[] and col[] respectively, consider the matrix as submatrix starting from second row and second column and apply method 1.  
4) Finally, using rowFlag and colFlag, update first row and first column if needed.

Time Complexity: O(M\*N)  
Auxiliary Space: O(1)

# Linked List

## Find first two numbers whose sum equals a given number in infinite length (stream of numbers) singly linked list.

Let the sum be S

Traverse the list and for each element X.

Seach if S-X exists in Hash, if yes X and S-X are the two numbers

If not Put X in Hash and continue.

## Given two lists, each containing numbers, how would you find the intersection of these two lists? What if these two lists are read from a huge file that cannot fit in memory?

If the lists are sorted:

Traverse both the lists in a single loop and keep on increasing their pointers based on the smaller number.

It not sorted:

Hash the first list and search the second list’s nodes in hash. If found add them to intersection.

// sort List1 and List 2 --> complexity = O(mlogm + nlogn)

int i, j;

i = 0; i = 0;

while(i < size of List1 and j < size of List 2)

{

if(A[i] == B[j])

{

print(A[i]);

i++;j++;

}

else if(A[i] > B[j])

{

i++;

}

else{

j++;

}

}

1. Create a B+ Tree from List1, where each leaf is different file(As the list is very large, and can't loaded to whole list to memory at once).  
2. For each element in List2, seach it from B+ tree(which is of List1).

T = O(M\*Log(N)(of base k) + N\*Log(N)(of base k).

N: Number of elements in List1.  
M: Number of elements in List2.  
k: B+ Tree is k-way tree.

## Find the intersection point of 2 linked list (without hashing at a lesser complexity preferably O(m+n)).

(the 2 nodes from the seperate lists point to a same node from which the list continues as a single list)

There are many methods given at: <http://www.geeksforgeeks.org/archives/2405>

Sol 1:

scan both the lists once. Max(m,n) is the first scan complexity - where m and n are lengths of linked lists. Then take the diff in lengths (m-n). Traverse through the longer list from its head till the diff (m-n). Start traversing now in the second list too. Both traversing pointers in bigger and smaller lists would meet at the intersection. So here we scan the lists only twice.

1) Get count of the nodes in first list, let count be c1.  
2) Get count of the nodes in second list, let count be c2.  
3) Get the difference of counts d = abs(c1 – c2)  
4) Now traverse the bigger list from the first node till d nodes so that from here onwards both the lists have equal no of nodes.  
5) Then we can traverse both the lists in parallel till we come across a common node. (Note that getting a common node is done by comparing the address of the nodes)

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>    /\* Link list node \*/  struct node  {  int data;  struct node\* next;  };    /\* Function to get the counts of node in a linked list \*/  int getCount(struct node\* head);    /\* function to get the intersection point of two linked  lists head1 and head2 where head1 has d more nodes than  head2 \*/  int \_getIntesectionNode(int d, struct node\* head1, struct node\* head2);    /\* function to get the intersection point of two linked  lists head1 and head2 \*/  int getIntesectionNode(struct node\* head1, struct node\* head2)  {  int c1 = getCount(head1);  int c2 = getCount(head2);  int d;    if(c1 > c2)  {  d = c1 - c2;  return \_getIntesectionNode(d, head1, head2);  }  else  {  d = c2 - c1;  return \_getIntesectionNode(d, head2, head1);  }  }    /\* function to get the intersection point of two linked  lists head1 and head2 where head1 has d more nodes than  head2 \*/  int \_getIntesectionNode(int d, struct node\* head1, struct node\* head2)  {  int i;  struct node\* current1 = head1;  struct node\* current2 = head2;    for(i = 0; i < d; i++)  {  if(current1 == NULL)  { return -1; }  current1 = current1->next;  }    while(current1 != NULL && current2 != NULL)  {  if(current1 == current2)  return current1->data;  current1= current1->next;  current2= current2->next;  }    return -1;  }    /\* Takes head pointer of the linked list and  returns the count of nodes in the list \*/  int getCount(struct node\* head)  {  struct node\* current = head;  int count = 0;    while (current != NULL)  {  count++;  current = current->next;  }    return count;  }    Sol 2: **Reverse the first list and make equations**  1) Let X be the length of the first linked list until intersection point.  Let Y be the length of the second linked list until the intersection point.  Let Z be the length of the linked list from intersection point to End of  the linked list including the intersection node.  We Have  X + Z = C1;  Y + Z = C2;  2) Reverse first linked list.  3) Traverse Second linked list. Let C3 be the length of second list - 1.  Now we have  X + Y = C3  We have 3 linear equations. By solving them, we get  X = (C1 + C3 – C2)/2;  Y = (C2 + C3 – C1)/2;  Z = (C1 + C2 – C3)/2;  WE GOT THE INTERSECTION POINT.  4) Reverse first linked list.  Advantage: No Comparison of pointers. Disadvantage : Modifying linked list(Reversing list). |

## Reversing of double linked list

## Reverse a linked list

**Iterative Procedure**

The following are the sequence of steps to be followed:

Initially take three pointers: PrevNode, CurrNode, NextNode

Let CurrNode point to HeaderNode of the list. And let PrevNode and NextNode points to null

Now iterate through the linked list until CurrNode is null

In the loop, we need to change NextNode to PrevNode, PrevNode to CurrNode and CurrNode to NextNode

public ListNode reverseList(ListNode headerNode)

{

ListNode prevNode = null;

ListNode currNode = headerNode;

ListNode nextNode = null;

while (currNode != null)

{

nextNode = currNode.next;

currNode.next = prevNode;

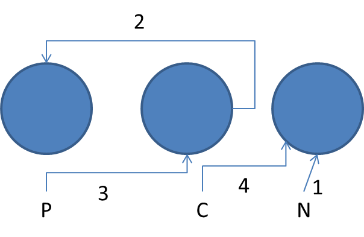
prevNode = currNode;

currNode = nextNode;

}

return prevNode;

}



**Recursive**

The following are the sequence of steps to be followed:

If the list is empty, then the reverse of the list is also empty

If the list has one element, then the reverse of the list is the element itself

If the list has n elements, then the reverse of the complete list is reverse of the list starting from second node followed by the first node element. This step is recursive step

public ListNode reverseList(ListNode headerNode)

{

// Reverse of a empty list or null list is null

if (headerNode == null)

{

return null;

}

// Reverse of a single element list is the list with that element

if (headerNode.next == null)

{

return headerNode;

}

// Reverse of n element list is reverse of the second element followed by first element

// Get the list node pointed by second element

ListNode secondElementNode = headerNode.next;

// Unlink the first element

headerNode.next = null;

// Reverse everything from the second element

ListNode revNode = reverseList(secondElementNode);

// Now we join both the lists

secondElementNode.next = headerNode;

return revNode;

}

## Pairwise swap elements of a given linked list

**Sol 1 Recursive:**

node\* pariWiseSwap(node \*\*head)

{

node \*first = \*head;

if(first && first->next)

{

node \*second = first->next;

first->next = pariWiseSwap(&second->next);

second->next = first;

first = second;

}

return first;

}

**Sol 2a Iterative:**

struct node{

int value;

struct node \*next;

};

typedef struct node\* NODEPTR;

PairwiseReverse(NODEPTR \*head){

NODEPTR first,second;

if(\*head == NULL)

return;

first = \*head;

second = first->next;

if(!second)

return;

first->next = second->next;

second->next = first;

\*head = second;

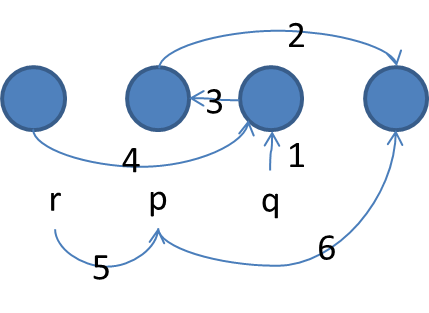
while( first->next && first->next->next ){

first=first->next;

second=first->next;

first->next = second->next;

second->next = first;

 }

}

**Sol 2b Iterative: Different coding**

//p & q points to nodes currently being swapped. And r holds the last node of previously swapped list.

p=head;

q=NULL;

r=NULL;

while(p!=NULL && p->next!=NULL)

{

q=p->next;

printf("Swapping %d and %d\n",p->value,q->value);

p->next=q->next;

q->next=p;

if(r!=NULL)

r->next=q;

if(p==head)

head=q;

r=p;

p=p->next;

}

## Reverse a Linked List in groups of given size K

**Sol 1:**

HeaderNode is the head of the linked list

Take three pointers StartNode, EndNode and NextNode

Let the NextNode pointer points to HeaderNode and unlink HeaderNode

Repeat the following steps until NextNode is null

Point StartNode and EndNode to NextNode

Move EndNode K nodes away from StartNode

Point NextNode to the node next to EndNode

Unlink EndNode from the linked list

Now reverse the list pointed by StartNode which gives reverse of K nodes

If HeaderNode is null point the HeaderNode to reversed list else point the reversed list to the end of the HeaderNode list

Hence the list pointed by HeaderNode contains the K- Reverse of a linked list

public ListNode reverseKListNodes(ListNode headerNode, int k)

{

// Take 3 pointers startNode, endNode, nextNode pointing to headerNode

ListNode nextNode = headerNode;

ListNode startNode = null;

ListNode endNode = null;

headerNode = null;

while (nextNode != null)

{

// startNode and endNode points to nextNode

startNode = nextNode;

endNode = nextNode;

// Move endNode pointing towards node after k elements from startNode

for (int i = 1; i < k; i++)

{

endNode = endNode.next;

if (endNode == null)

{

break;

}

}

// If endNode is not null, then reverse the list starting from startNode to endNode

// else if endNode is null, then there is nothing to reverse

if (endNode != null)

{

// Save the node next to endNode

nextNode = endNode.next;

// Unlink the endNode

endNode.next = null;

// Reverse the list starting from startNode

ReverseSingleListIterative reverseListIter = new ReverseSingleListIterative();

startNode = reverseListIter.reverseList(startNode);

}

else

{

nextNode = null;

}

// Point headerNode to the startNode of the first iteration.

// If the headerNode is set, append the list startNode to the headerNode

if (headerNode == null)

{

headerNode = startNode;

}

else

{

SingleLinkedList.getLastNode(headerNode).next = startNode;

}

}

return headerNode;

}

**Sol 2**

<http://www.geeksforgeeks.org/archives/8014>

Note: In this solution all the strands are reversd; even if last strand is less than K it is reversed.

struct node \*reverseK (struct node \*head, int k)

{

struct node\* current = head;

struct node\* next;

struct node\* prev = NULL;

int count = 0;

/\*reverse first k nodes of the linked list \*/

while (current != NULL && count < k)

{

next = current->next;

current->next = prev;

prev = current;

current = next;

count++;

}

/\* next is now a pointer to (k+1)th node

Recursively call for the list starting from current.

And make rest of the list as next of first node \*/

if(next != NULL) // CHECK shoud it be next or current;

{ head->next = reverseK(next, k); }

/\* prev is new head of the input list \*/

return prev;

}

## Given multiple sorted lists, merge them into a new sorted list.

Sol1: Take the minimum element from the lists and add it to the new list. Increment the pointer of the list from which we have taken the element.

O(nk)

Sol 2: O(n lgK)

Suppose we have total K sorted linked lists.  
And total no of element is N.  
Now created a single min heap of size K (with the first element of each linked list)  
Extract Min and add it to final list.  
Add the next node of extracted node in heap (if it is not null).  
Repeat for all element.

Time : O(N\*lnK);  
space : O(N)

## Input linked list:

1->2->3->4->a->b->c->d->5->6->e->f

Output should be in below format :

1->a->2->b->3->c->4->d->5->e->6->f

Convert it into two linked lists. One containing only numbers and other only characters

Then we can do a merge of these two lists.

Time Complexity O(n)  
Space Complexity O(1)

## Find the kth element from last in link list

LinkedListNode nthToLast(LinkedListNode head, int n) {

if (head == null || n < 1) {

return null;

}

LinkedListNode p1 = head;

LinkedListNode p2 = head;

for (int j = 0; j < n ; ++j) { // skip n steps ahead

if (p2 == null) {

return null; // not found since list size < n

}

p2 = p2.next;

}

while (p2 != null) {

p1 = p1.next;

p2 = p2.next;

}

return p1;

}

## You are given a singly linked list. On seeing a node, if it has a node with greater value than itself on its right delete the seen node. Return the head of the result singly linked list.

Can be done in O(n)

First reverse the linked list and then treverse the linked list.

While traversing maintain the max value

If you find that any node value is less than max delete it.

After this again reverse the linked list and return it.

struct node{node \*next; int data; };

struct node \*DeleteGreater(struct node \*head)

{

struct node \*temp=NULL;

struct node \*temp1=NULL;

int max=-32767;

if(!head||head->next==NULL)

return head;

head=reverse(head);

temp=head;

max=temp->data;

while(temp->next)

{

if(temp->next->data < max)

{

temp1=temp->next;

temp->next=temp->next->next;

free(temp1);

}

else

{

max=temp->next->data;

temp=temp->next;

}

}

head=reverse(head);

return head;

}

## Sort a doubly link list in less then O(nlogn) time

Quick sort. but its worst case is O(n^2)

Merge sort. Merge sort is also good for single linked list.

Let head be the first node of the linked list to be sorted and headRef be the pointer to head. Note that we need a reference to head in MergeSort() as the below implementation changes next links to sort the linked lists (not data at the nodes), so head node has to be changed if the data at original head is not the smallest value in linked list.

MergeSort(headRef)

1) If head is NULL or there is only one element in the Linked List

then return.

2) Else divide the linked list into two halves.

FrontBackSplit(head, &a, &b); /\* a and b are two halves \*/

3) Sort the two halves a and b.

MergeSort(a);

MergeSort(b);

4) Merge the sorted a and b (using SortedMerge() discussed [here](http://geeksforgeeks.org/?p=3622))

and update the head pointer using headRef.

\*headRef = SortedMerge(a, b);

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  /\* Link list node \*/  struct node  {  int data;  struct node\* next;  };  /\* function prototypes \*/  struct node\* SortedMerge(struct node\* a, struct node\* b);  void FrontBackSplit(struct node\* source, struct node\*\* frontRef, struct node\*\* backRef);  /\* sorts the linked list by changing next pointers (not data) \*/  void MergeSort(struct node\*\* headRef)  {  struct node\* head = \*headRef;  struct node\* a;  struct node\* b;  /\* Base case -- length 0 or 1 \*/  if ((head == NULL) || (head->next == NULL))  {  return;  }  /\* Split head into 'a' and 'b' sublists \*/  FrontBackSplit(head, &a, &b);  /\* Recursively sort the sublists \*/  MergeSort(&a);  MergeSort(&b);  /\* answer = merge the two sorted lists together \*/  \*headRef = SortedMerge(a, b);  }  /\* See http://geeksforgeeks.org/?p=3622 for details of this  function \*/  struct node\* SortedMerge(struct node\* a, struct node\* b)  {  struct node\* result = NULL;  /\* Base cases \*/  if (a == NULL)  return(b);  else if (b==NULL)  return(a);  /\* Pick either a or b, and recur \*/  if (a->data <= b->data)  {  result = a;  result->next = SortedMerge(a->next, b);  }  else  {  result = b;  result->next = SortedMerge(a, b->next);  }  return(result);  }  /\* UTILITY FUNCTIONS \*/  /\* Split the nodes of the given list into front and back halves,  and return the two lists using the reference parameters.  If the length is odd, the extra node should go in the front list.  Uses the fast/slow pointer strategy. \*/  void FrontBackSplit(struct node\* source,  struct node\*\* frontRef, struct node\*\* backRef)  {  struct node\* fast;  struct node\* slow;  if (source==NULL || source->next==NULL)  {  /\* length < 2 cases \*/  \*frontRef = source;  \*backRef = NULL;  }  else  {  slow = source;  fast = source->next;  /\* Advance 'fast' two nodes, and advance 'slow' one node \*/  while (fast != NULL)  {  fast = fast->next;  if (fast != NULL)  {  slow = slow->next;  fast = fast->next;  }  }  /\* 'slow' is before the midpoint in the list, so split it in two  at that point. \*/  \*frontRef = source;  \*backRef = slow->next;  slow->next = NULL;  }  }  /\* Function to print nodes in a given linked list \*/  void printList(struct node \*node)  {  while(node!=NULL)  {  printf("%d ", node->data);  node = node->next;  }  }  /\* Function to insert a node at the beginging of the linked list \*/  void push(struct node\*\* head\_ref, int new\_data)  {  /\* allocate node \*/  struct node\* new\_node =  (struct node\*) malloc(sizeof(struct node));  /\* put in the data \*/  new\_node->data = new\_data;  /\* link the old list off the new node \*/  new\_node->next = (\*head\_ref);  /\* move the head to point to the new node \*/  (\*head\_ref) = new\_node;  }  /\* Drier program to test above functions\*/  int main()  {  /\* Start with the empty list \*/  struct node\* res = NULL;  struct node\* a = NULL;  /\* Let us create a unsorted linked lists to test the functions  Created lists shall be a: 2->3->20->5->10->15 \*/  push(&a, 15);  push(&a, 10);  push(&a, 5);  push(&a, 20);  push(&a, 3);  push(&a, 2);  /\* Sort the above created Linked List \*/  MergeSort(&a);  printf("\n Sorted Linked List is: \n");  printList(a);  getchar();  return 0;  } |

Time Complexity: O(nLogn)

## Random pointer is present in every node of the linked list . And they will be pointing to any of the node of the list . We need to clone this list and return it .

1. clone the list without considering random pointers. For each node, add an entry

to a hashmap: old\_addr -> clone\_addr

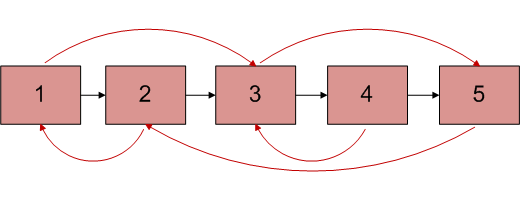
2. use the hashmap to set up the random pointers

Other solutions:

## Copy a linked list with next and arbit pointer

You are given a Double Link List with one pointer of each node pointing to the next node just like in a single link list. The second pointer however CAN point to any node in the list and not just the previous node. Now write a program in **O(n) time** to duplicate this list. That is, write a program which will create a copy of this list.

Let us call the second pointer as arbit pointer as it can point to any arbitrary node in the linked list.



Arbitrary pointers are shown in red and next pointers in black

Figure 1

**Method 1 (Uses O(n) extra space)**  
This method stores the next and arbitrary mappings (of original list) in an array first, then modifies the original Linked List (to create copy), creates a copy. And finally restores the original list.

1) Create all nodes in copy linked list using next pointers.  
3) Store the node and its next pointer mappings of original linked list.  
3) Change next pointer of all nodes in original linked list to point to the corresponding node in copy linked list.  
Following diagram shows status of both Linked Lists after above 3 steps. The red arrow shows arbit pointers and black arrow shows next pointers.

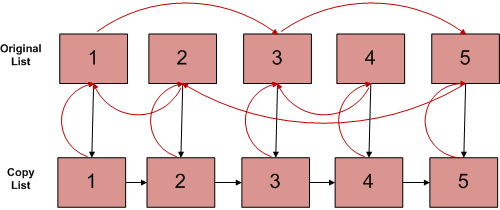


Figure 2

4) Change the arbit pointer of all nodes in copy linked list to point to corresponding node in original linked list.  
5) Now construct the arbit pointer in copy linked list as below and restore the next pointer of nodes in the original linked list.

copy\_list\_node->arbit =

copy\_list\_node->arbit->arbit->next;

copy\_list\_node = copy\_list\_node->next;

6) Restore the next pointers in original linked list from the stored mappings(in step 2).

Time Complexity:  O(n)  
Auxiliary Space:  O(n)

**Method 2 (Uses Constant Extra Space)**  
1) Create the copy of node 1 and insert it between node 1 & node 2 in original Linked List, create the copy of 2 and insert it between 2 & 3.. Continue in this fashion, add the copy of N afte the Nth node  
2) Now copy the arbitrary link in this fashion

original->next->arbitrary = original->arbitrary->next; /\*TRAVERSE

TWO NODES\*/

This works because original->next is nothing but copy of original and Original->arbitrary->next is nothing but copy of arbitrary.  
3) Now restore the original and copy linked lists in this fashion in a single loop.

original->next = original->next->next;

copy->next = copy->next->next;

4) Make sure that last element of original->next is NULL.

Time Complexity: O(n)  
Auxiliary Space: O(1)

## Merge 2 sorted linked list into another sorted linked list, then do it in O(N) time and

1. Take 4 pointers, temp1 traverser for t1

temp2 - taraverser for t2

head - head of the new list

temp - end node of new list

2. compare temp1 temp2 and copy in new list, till either list ends

3. Link the remaining list of temp1 or temp2 to new list

// merging the two sorted lionked list

struct node \*Merge2linkedlist(struct node \*l1,struct node \*l2)

{

struct node \*temp1,\*temp2,\*temp,\*head = NULL;

if(!l1)

return l2;

if(!l2)

return l1;

temp1=l1;

temp2=l2;

while(temp1 && temp2)

{

if(temp1->info<=temp2->info)

{

if(!head)

{

head=temp1;

temp=head;

temp->next=NULL;

temp1=temp1->next;

}

else

{

temp->next=temp1;

temp=temp->next;

temp->next=NULL;

temp1=temp1->next;

}

}

else

{

if(!head)

{

head=temp2;

temp=temp2;

temp->next=NULL;

temp2=temp2->next;

}

else

{

temp->next=temp2;

temp=temp->next;

temp->next=NULL;

temp2=temp2->next;

}

}

}//end while

//Link the remining list to new list

if(temp1==NULL && temp2!=NULL)

{

temp->next=temp2;

}

if(temp2==NULL && temp1!=NULL)

{

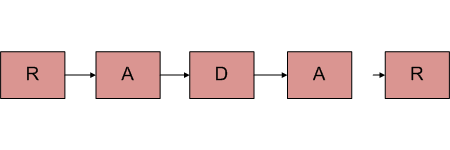
temp->next=temp1;

}

return head;

}

## How do you test the numbers present in a linked list is palindrome or not? (List is a singly linked list)

Asked by Varun Bhatia.  
  


**METHOD 1 (By reversing the list)**

1. Get the middle of the linked list.

2. Reverse the second half of the linked list.

3. Compare the first half and second half.

4. Construct the original linked list by reversing the

second half again and attaching it back to the first half

Implementation:

/\* Program to check if a linked list is palindrome \*/

#include<stdio.h>

#include<stdlib.h>

#define bool int

/\* Link list node \*/

struct node

{

char data;

struct node\* next;

};

void reverse(struct node\*\*);

bool compareLists(struct node\*, struct node \*);

/\* Function to check if given linked list is

palindrome or not \*/

bool isPalindrome(struct node \*head)

{

struct node \*slow\_ptr = head;

struct node \*fast\_ptr = head;

struct node \*second\_half;

struct node \*prev\_of\_slow\_ptr = head;

char res;

if(head!=NULL)

{

/\* Get the middle of the list. Move slow\_ptr by 1

and fast\_ptrr by 2, slow\_ptr will have the |\_n/2\_|th

node \*/

while((fast\_ptr->next)!=NULL &&

(fast\_ptr->next->next)!=NULL)

{

fast\_ptr = fast\_ptr->next->next;

/\*We need previous of the slow\_ptr for

linked lists with odd elements \*/

prev\_of\_slow\_ptr = slow\_ptr;

slow\_ptr = slow\_ptr->next;

}

/\* Case where we have even no of elements \*/

if(fast\_ptr->next != NULL)

{

second\_half = slow\_ptr->next;

reverse(&second\_half);

slow\_ptr->next = NULL;

res = compareLists(head, second\_half);

/\*construct the original list back\*/

reverse(&second\_half);

slow\_ptr->next = second\_half;

}

/\* Case where we have odd no. of elements. Neither first

nor second list should have the middle element \*/

else

{

second\_half = slow\_ptr->next;

prev\_of\_slow\_ptr->next = NULL;

reverse(&second\_half);

res = compareLists(head, second\_half);

/\*construct the original list back\*/

reverse(&second\_half);

prev\_of\_slow\_ptr->next = slow\_ptr;

slow\_ptr->next = second\_half;

}

return res;

}

}

/\* Function to reverse the linked list Note that this

function may change the head \*/

void reverse(struct node\*\* head\_ref)

{

struct node\* prev = NULL;

struct node\* current = \*head\_ref;

struct node\* next;

while (current != NULL)

{

next = current->next;

current->next = prev;

prev = current;

current = next;

}

\*head\_ref = prev;

}

/\* Function to check if two input lists have same data\*/

int compareLists(struct node\* head1, struct node \*head2)

{

struct node\* temp1 = head1;

struct node\* temp2 = head2;

while(temp1 && temp2)

{

if(temp1->data == temp2->data)

{

temp1 = temp1->next;

temp2 = temp2->next;

}

else return 0;

}

/\* Both are empty reurn 1\*/

if(temp1 == NULL && temp2 == NULL)

return 1;

/\* Will reach here when one is NULL

and other is not \*/

return 0;

}

/\* Push a node to linked list. Note that this function

changes the head \*/

void push(struct node\*\* head\_ref, char new\_data)

{

/\* allocate node \*/

struct node\* new\_node =

(struct node\*) malloc(sizeof(struct node));

/\* put in the data \*/

new\_node->data = new\_data;

/\* link the old list off the new node \*/

new\_node->next = (\*head\_ref);

/\* move the head to pochar to the new node \*/

(\*head\_ref) = new\_node;

}

/\* Drier program to test above function\*/

int main()

{

/\* Start with the empty list \*/

struct node\* head = NULL;

push(&head, 'p');

push(&head, 'e');

push(&head, 'e');

push(&head, 'p');

/\* p->e->e->p \*/

if(isPalindrome(head) == 1)

printf("Linked list is Palindrome");

else

printf("Linked list is not Palindrome");

getchar();

return 0;

}

Time Complexity O(n)  
Auxiliary Space: O(1)

**METHOD 2 (Using Recursion)**  
Use two pointers left and right. Move right and left using recursion and check for following in each recursive call.  
1) Sub-list is palindrome.  
2) Value at current left and right are matching.

If both above conditions are true then return true.

#define bool int

#include<stdio.h>

#include<stdlib.h>

/\* Link list node \*/

struct node

{

char data;

struct node\* next;

};

bool isPalindrome(struct node \*\*left, struct node \*right)

{

/\* stop recursion here \*/

if (!right)

return true;

/\* If sub-list is not palindrome then no need to

check for current left and right, return false \*/

bool isp = isPalindrome(left, right->next);

if (isp == false)

return false;

/\* Check values at current left and right \*/

bool isp1 = (right->data == (\*left)->data);

/\* Move left to next node \*/

\*left = (\*left)->next; /\* save next pointer \*/

return isp1;

}

/\* UTILITY FUNCTIONS \*/

/\* Push a node to linked list. Note that this function

changes the head \*/

void push(struct node\*\* head\_ref, char new\_data)

{

/\* allocate node \*/

struct node\* new\_node =

(struct node\*) malloc(sizeof(struct node));

/\* put in the data \*/

new\_node->data = new\_data;

/\* link the old list off the new node \*/

new\_node->next = (\*head\_ref);

/\* move the head to pochar to the new node \*/

(\*head\_ref) = new\_node;

}

/\* Drier program to test above function\*/

int main()

{

/\* Start with the empty list \*/

struct node\* head = NULL;

push(&head, 'r');

push(&head, 'a');

push(&head, 'd');

push(&head, 'a');

push(&head, 'r');

/\* r->a->d->a->r\*/

if(isPalindrome(&head, head) == 1)

printf("Linked list is Palindrome");

else

printf("Linked list is not Palindrome");

getchar();

return 0;

}

Time Complexity: O(n)  
Auxiliary Space: O(n) if Function Call Stack size is considered, otherwise O(1).

## Convert the matrix into a linked list (singly or doubly). Nodes have two pointers - right and down.

a1-a2-a3-a4

| | | |

b1-b2-b3-b4

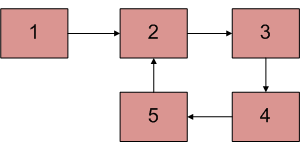
| | | |

c1-c2-c3-c4

<http://www.careercup.com/question?id=12364850>

## Detect and Remove Loop in a Linked List

Write a function detectAndRemoveLoop() that checks whether a given Linked List contains loop and if loop is present then removes the loop and returns true. And if the list doesn’t contain loop then returns false. Below diagram shows a linked list with a loop. detectAndRemoveLoop() must change the below list to 1->2->3->4->5->NULL.



We recommend to read following post as a prerequisite.  
 [Write a C function to detect loop in a linked list](http://geeksforgeeks.org/?p=112)

Before trying to remove the loop, we must detect it. Techniques discussed in the above post can be used to detect loop. To remove loop, all we need to do is to get pointer to the last node of the loop. For example, node with value 5 in the above diagram. Once we have pointer to the last node, we can make the next of this node as NULL and loop is gone.  
We can easily use Hashing or Visited node techniques (discussed in the aobve mentioned post) to get the pointer to the last node. Idea is simple: the very first node whose next is already visited (or hashed) is the last node.  
We can also use Floyd Cycle Detection algorithm to detect and remove the loop. In the Floyd’s algo, the slow and fast pointers meet at a loop node. We can use this loop node to remove cycle. There are following two different ways of removing loop when Floyd’s algorithm is used for Loop detection.

**Method 1 (Check one by one)**   
We know that Floyd’s Cycle detection algorithm terminates when fast and slow pointers meet at a common point. We also know that this common point is one of the loop nodes (2 or 3 or 4 or 5 in the above diagram). We store the address of this in a pointer variable say ptr2. Then we start from the head of the Linked List and check for nodes one by one if they are reachable from ptr2. When we find a node that is reachable, we know that this node is the starting node of the loop in Linked List and we can get pointer to the previous of this node.

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>    /\* Link list node \*/  struct node  {      int data;      struct node\* next;  };    /\* Function to remove loop. Used by detectAndRemoveLoop() \*/  void removeLoop(struct node \*, struct node \*);    /\* This function detects and removes loop in the list    If loop was there in the list then it returns 1,    otherwise returns 0 \*/  int detectAndRemoveLoop(struct node \*list)  {      struct node  \*slow\_p = list, \*fast\_p = list;        while (slow\_p && fast\_p && fast\_p->next)      {          slow\_p = slow\_p->next;          fast\_p  = fast\_p->next->next;            /\* If slow\_p and fast\_p meet at some point then there             is a loop \*/          if (slow\_p == fast\_p)          {              removeLoop(slow\_p, list);                /\* Return 1 to indicate that loop is found \*/              return 1;          }      }        /\* Return 0 to indeciate that ther is no loop\*/      return 0;  }    /\* Function to remove loop.   loop\_node --> Pointer to one of the loop nodes   head -->  Pointer to the start node of the linked list \*/  void removeLoop(struct node \*loop\_node, struct node \*head)  {     struct node \*ptr1;     struct node \*ptr2;       /\* Set a pointer to the beging of the Linked List and        move it one by one to find the first node which is        part of the Linked List \*/     ptr1 = head;     while(1)     {       /\* Now start a pointer from loop\_node and check if it ever         reaches ptr2 \*/       ptr2 = loop\_node;       while(ptr2->next != loop\_node && ptr2->next != ptr1)       {           ptr2 = ptr2->next;       }         /\* If ptr2 reahced ptr1 then there is a loop. So break the          loop \*/       if(ptr2->next == ptr1)          break;         /\* If ptr2 did't reach ptr1 then try the next node after ptr1 \*/       else         ptr1 = ptr1->next;     }       /\* After the end of loop ptr2 is the last node of the loop. So       make next of ptr2 as NULL \*/     ptr2->next = NULL;  }    /\* UTILITY FUNCTIONS \*/  /\* Given a reference (pointer to pointer) to the head    of a list and an int, pushes a new node on the front    of the list. \*/  void push(struct node\*\* head\_ref, int new\_data)  {      /\* allocate node \*/      struct node\* new\_node =          (struct node\*) malloc(sizeof(struct node));        /\* put in the data  \*/      new\_node->data  = new\_data;        /\* link the old list off the new node \*/      new\_node->next = (\*head\_ref);        /\* move the head to point to the new node \*/      (\*head\_ref)    = new\_node;  }    /\* Function to print linked list \*/  void printList(struct node \*node)  {      while(node != NULL)      {          printf("%d  ", node->data);          node = node->next;      }  }    /\* Drier program to test above function\*/  int main()  {      /\* Start with the empty list \*/      struct node\* head = NULL;        push(&head, 10);      push(&head, 4);      push(&head, 15);      push(&head, 20);      push(&head, 50);        /\* Create a loop for testing \*/      head->next->next->next->next->next = head->next->next;        detectAndRemoveLoop(head);        printf("Linked List after removing loop \n");      printList(head);        getchar();      return 0;  } |

**Method 2 (Efficient Solution)**  
This method is also dependent on Floyd’s Cycle detection algorithm.  
1) Detect Loop using Floyd’s Cycle detection algo and get the pointer to a loop node.  
2) Count the number of nodes in loop. Let the count be k.  
3) Fix one pointer to the head and another to kth node from head.  
4) Move both pointers at the same pace, they will meet at loop starting node.  
5) Get pointer to the last node of loop and make next of it as NULL.

Thanks to WgpShashank for suggesting this method.

#include<stdio.h>

#include<stdlib.h>

/\* Link list node \*/

struct node

{

int data;

struct node\* next;

};

/\* Function to remove loop. \*/

void removeLoop(struct node \*, struct node \*);

/\* This function detects and removes loop in the list

If loop was there in the list then it returns 1,

otherwise returns 0 \*/

int detectAndRemoveLoop(struct node \*list)

{

struct node \*slow\_p = list, \*fast\_p = list;

while (slow\_p && fast\_p && fast\_p->next)

{

slow\_p = slow\_p->next;

fast\_p = fast\_p->next->next;

/\* If slow\_p and fast\_p meet at some point then there

is a loop \*/

if (slow\_p == fast\_p)

{

removeLoop(slow\_p, list);

/\* Return 1 to indicate that loop is found \*/

return 1;

}

}

/\* Return 0 to indeciate that ther is no loop\*/

return 0;

}

/\* Function to remove loop.

loop\_node --> Pointer to one of the loop nodes

head --> Pointer to the start node of the linked list \*/

void removeLoop(struct node \*loop\_node, struct node \*head)

{

struct node \*ptr1 = loop\_node;

struct node \*ptr2 = loop\_node;

// Count the number of nodes in loop

unsigned int k = 1, i;

while (ptr1->next != ptr2)

{

ptr1 = ptr1->next;

k++;

}

// Fix one pointer to head

ptr1 = head;

// And the other pointer to k nodes after head

ptr2 = head;

for(i = 0; i < k; i++)

ptr2 = ptr2->next;

/\* Move both pointers at the same pace,

they will meet at loop starting node \*/

while(ptr2 != ptr1)

{

ptr1 = ptr1->next;

ptr2 = ptr2->next;

}

// Get pointer to the last node

ptr2 = ptr2->next;

while(ptr2->next != ptr1)

ptr2 = ptr2->next;

/\* Set the next node of the loop ending node

to fix the loop \*/

ptr2->next = NULL;

}

/\* UTILITY FUNCTIONS \*/

/\* Given a reference (pointer to pointer) to the head

of a list and an int, pushes a new node on the front

of the list. \*/

void push(struct node\*\* head\_ref, int new\_data)

{

/\* allocate node \*/

struct node\* new\_node =

(struct node\*) malloc(sizeof(struct node));

/\* put in the data \*/

new\_node->data = new\_data;

/\* link the old list off the new node \*/

new\_node->next = (\*head\_ref);

/\* move the head to point to the new node \*/

(\*head\_ref) = new\_node;

}

/\* Function to print linked list \*/

void printList(struct node \*node)

{

while(node != NULL)

{

printf("%d ", node->data);

node = node->next;

}

}

/\* Drier program to test above function\*/

int main()

{

/\* Start with the empty list \*/

struct node\* head = NULL;

push(&head, 10);

push(&head, 4);

push(&head, 15);

push(&head, 20);

push(&head, 50);

/\* Create a loop for testing \*/

head->next->next->next->next->next = head->next->next;

detectAndRemoveLoop(head);

printf("Linked List after removing loop \n");

printList(head);

getchar();

return 0;

}

## How to find out middle element from a looped single linked list

Find the start node of the loop ptr2.

Now move two pointers slow and fast.

When fast->next or fast->next->next is ptr2 slow is the middle.

## Find the middle element in a singlely linked list

## How will you add two nos represented as a singly linked list ( 2->9->1->7+5->1->7= 3->4->3->4 )

**Sol 1: Involves reversing the list.**

reverse list-1

reverse list-2

find the sum and store it in a new list represented by list-3

reverse the list.

to add two numbers represented by two linked list just reverse the two linked list first and in a loop just add two linked list numbers node by node taking care of carry and just using the malloc function make the third linked list reversing the linked list u can do it in O(n) time using any of recusrion or iterative scheme

and reversing the doubly linked list just swap the two pointers of each node thaat is swap the prev and next pointer of each node while traversing this will take the O(n) time where n is the length of DLL

**Sol 2: Without reversing the lists**

[**http://stackoverflow.com/questions/7294048/add-two-big-numbers-represented-as-linked-lists-without-reversing-the-linked-list**](http://stackoverflow.com/questions/7294048/add-two-big-numbers-represented-as-linked-lists-without-reversing-the-linked-list)

1.first traverse both list and find out their length

2.find the difference in their lenngth.

3.if both are of same length then its ok point pointers ptr1 and ptr2 at the begining of the list

4.otherwise

traverse the larger list by the amount equal to the difference between the length in advance and keep a pointer to point that position say ptr1.

keep a pointer ptr2 pointing at other list

5.call a function say int Add\_List(ptr1,ptr2) which looks like:

int Add\_List (node \*ptr1,node \*ptr2)

{

if(ptr1!=NULL)

{

int sum=Add\_List(ptr1-next,ptr2->next);

sum=sum+ptr1->info+ptr2->info;

if(sum>=10)

{

int mod=sum%10;

ptr1->info=mod;//when sum is > or = 10 then mod will be 1 which will be carry for the previous one so returning 1

return 1;

}

else

{

ptr1->info=sum;

return 0;

}

}

else

return 0;//this statement will be exucuted only once when ptr1==NULL and will return 0 which will initialise sum to 0

}

At last this function will return either 1 or 0 to the function that called it.

Now this returned value is carry for the last number in the remaining part of the larger list,remember we already moved by a difference in the starting.

Again for that part add this carry with the help of ur own function and finally return to the main with carry of the the very first element.

If returned carry in the main is 0 then ur larger list contains the desired sum otherwise create a new node with 1 as its info part and add it to the start of the larger list which is finally the requred sum.

# Trees, Binary Trees, and BST

## Deserialize a Binary Tree from it's Pre & In order traversals

Inorder sequence: D B E A F C  
Preorder sequence: A B D E C F

In a Preorder sequence, leftmost element is the root of the tree. So we know ‘A’ is root for given sequences. By searching ‘A’ in Inorder sequence, we can find out all elements on left side of ‘A’ are in left subtree and elements on right are in right subtree. So we know below structure now.

A

/ \

/ \

D B E F C

We recursively follow above steps and get the following tree.

A

/ \

/ \

B C

/ \ /

/ \ /

D E F

Algorithm: buildTree()  
1) Pick an element from Preorder. Increment a Preorder Index Variable (preIndex in below code) to pick next element in next recursive call.  
2) Create a new tree node tNode with the data as picked element.  
3) Find the picked element’s index in Inorder. Let the index be inIndex.  
4) Call buildTree for elements before inIndex and make the built tree as left subtree of tNode.  
5) Call buildTree for elements after inIndex and make the built tree as right subtree of tNode.  
6) return tNode.

/\* program to construct tree using inorder and preorder traversals \*/

#include<stdio.h>

#include<stdlib.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node

{

char data;

struct node\* left;

struct node\* right;

};

/\* Prototypes for utility functions \*/

int search(char arr[], int strt, int end, char value);

struct node\* newNode(char data);

/\* Recursive function to construct binary of size len from

Inorder traversal in[] and Preorder traversal pre[]. Initial values

of inStrt and inEnd should be 0 and len -1. The function doesn't

do any error checking for cases where inorder and preorder

do not form a tree \*/

struct node\* buildTree(char in[], char pre[], int inStrt, int inEnd)

{

static int preIndex = 0;

if(inStrt > inEnd)

return NULL;

/\* Pick current node from Preorder traversal using preIndex

and increment preIndex \*/

struct node \*tNode = newNode(pre[preIndex++]);

/\* If this node has no children then return \*/

if(inStrt == inEnd)

return tNode;

/\* Else find the index of this node in Inorder traversal \*/

int inIndex = search(in, inStrt, inEnd, tNode->data);

/\* Using index in Inorder traversal, construct left and

right subtress \*/

tNode->left = buildTree(in, pre, inStrt, inIndex-1);

tNode->right = buildTree(in, pre, inIndex+1, inEnd);

return tNode;

}

/\* UTILITY FUNCTIONS \*/

/\* Function to find index of value in arr[start...end]

The function assumes that value is present in in[] \*/

int search(char arr[], int strt, int end, char value)

{

int i;

for(i = strt; i <= end; i++)

{

if(arr[i] == value)

return i;

}

}

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(char data)

{

struct node\* node = (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

/\* This funtcion is here just to test buildTree() \*/

void printInorder(struct node\* node)

{

if (node == NULL)

return;

/\* first recur on left child \*/

printInorder(node->left);

/\* then print the data of node \*/

printf("%c ", node->data);

/\* now recur on right child \*/

printInorder(node->right);

}

/\* Driver program to test above functions \*/

int main()

{

char in[] = {'D', 'B', 'E', 'A', 'F', 'C'};

char pre[] = {'A', 'B', 'D', 'E', 'C', 'F'};

int len = sizeof(in)/sizeof(in[0]);

struct node \*root = buildTree(in, pre, 0, len - 1);

/\* Let us test the built tree by printing Insorder traversal \*/

printf("\n Inorder traversal of the constructed tree is \n");

printInorder(root);

getchar();

}

## Find Height of a tree without using recursion?

<http://www.geeksforgeeks.org/archives/2686>

public static int getHeightOfBTUsingLevelOrder(BTNode btRootNode){

if(btRootNode == null) return 0;

Queue queue = LLQueue.createQueue();

queue.enQueue(btRootNode);

queue.enQueue(sentinal);

int level = 0;

while(!queue.isEmpty()){

Object tempNode = queue.deQueue();

if(tempNode == sentinal){

if(!queue.isEmpty())

queue.enQueue(sentinal);

level++;

}else{

BTNode tempBTNode = (BTNode) tempNode;

if(tempBTNode.getLeft() != null){

queue.enQueue(tempBTNode.getLeft());

}

if(tempBTNode.getRight() != null){

queue.enQueue(tempBTNode.getRight());

}

}

}

return level;

}

***C++ Implementation***

int Height(Tree t) {

int height = -1;

if(t != NULL) {

std::list<Tree> q; //Queue to store tree nodes

q.push\_back(t);

q.push\_back(NULL); //null is the delimeter to show end of the level

while(!q.empty()) {

TreeNode \*node = q.front();

q.pop\_front();

if(node == NULL) {//delimeter encountered, increase height and push NULL if q not empty

height++;

if(!q.empty())

q.push\_back(NULL);

}

else {

if(node->left)

q.push\_back(node->left);

if(node->right)

q.push\_back(node->right);

}

}

}

return height;

}

## Given a binary tree, write a code that returns the difference between sum of nodes at even level and sum of nodes at odd level. Root is considered at level 0.

int DiffEvenOddLevels(Tree \*node)

{

if(!node)

return 0;

Q.enque(node);

Q.enque(terminal);

bool flag = true;

int sumEven = 0, sumOdd = 0;

while(Q.dequeCheck() != terminal)

{

for(Tree\* t = Q.deque(); t!=terminal; t=Q.deque())

{

if(flag) sumEven += t.val;

else sumOdd += t.val;

if(t->left) Q.enque(t->left);

if(t->right) Q.enque(t->right);

}

Q.enque(terminal);

flag!=flag;

}

return sumEven-sumOdd;

}

**Sol 2:**

int diffBetLevel(node\* root)

{

int lvalue=0, rvalue=0;

if(root==0)

return 0;

lvalue=diffBetLevel(root->left);

rvalue=diffBetLevel(root->right);

return root->data-(lvalue+rvalue);

}

## Given an N-Ary tree, WAP to find the minimum depth of the tree.

struct Node;

typedef std::list<Node\*> NodeList;

struct Node {

int data;

NodeList children;

}

int findMinDepth( Node\* root ) {

if ( !root || root->children.empty() ) return 0;

int minDepth = MAXINT;

for ( NodeList::const\_iterator

it = root->children.begin();

it != root->children.end();

++it ) {

int depth = findMinDepth(\*it);

if ( depth < minDepth ) minDepth = depth;

}

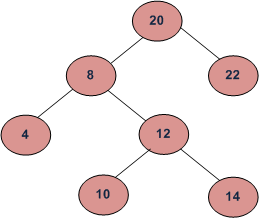
return 1 + minDepth;

}

## Given a Binary Search Tree, write a program to print the kth smallest element without using any static/global variable.

Given root of binary search tree and K as input, find K-th smallest element in BST. [Related Post](http://geeksforgeeks.org/forum/topic/adobe-interview-question-for-software-engineerdeveloper-fresher-3)

For example, in the following BST, if k = 3, then output should be 10, and if k = 5, then output should be 14.

[](http://geeksforgeeks.org/wp-content/uploads/BST.gif)

**Method 1: Using Inorder Traversal.**

Inorder traversal of BST retrieves elements of tree in the sorted order. The inorder traversal uses stack to store to be explored nodes of tree (threaded tree avoids stack and recursion for traversal, see [this post](http://geeksforgeeks.org/?p=6358)). The idea is to keep track of popped elements which participate in the order statics. Hypothetical algorithm is provided below,

Time complexity: O(n) where n is total nodes in tree..

**Algorithm:**

/\* initialization \*/

pCrawl = root

set initial stack element as NULL (sentinal)

/\* traverse upto left extreme \*/

while(pCrawl is valid )

stack.push(pCrawl)

pCrawl = pCrawl.left

/\* process other nodes \*/

while( pCrawl = stack.pop() is valid )

stop if sufficient number of elements are popped.

if( pCrawl.right is valid )

pCrawl = pCrawl.right

while( pCrawl is valid )

stack.push(pCrawl)

pCrawl = pCrawl.left

**Implementation:**

#include <stdio.h>

#include <stdlib.h>

#define ARRAY\_SIZE(arr) sizeof(arr)/sizeof(arr[0])

/\* just add elements to test \*/

/\* NOTE: A sorted array results in skewed tree \*/

int ele[] = { 20, 8, 22, 4, 12, 10, 14 };

/\* same alias \*/

typedef struct node\_t node\_t;

/\* Binary tree node \*/

struct node\_t

{

int data;

node\_t\* left;

node\_t\* right;

};

/\* simple stack that stores node addresses \*/

typedef struct stack\_t stack\_t;

/\* initial element always NULL, uses as sentinal \*/

struct stack\_t

{

node\_t\* base[ARRAY\_SIZE(ele) + 1];

int stackIndex;

};

/\* pop operation of stack \*/

node\_t \*pop(stack\_t \*st)

{

node\_t \*ret = NULL;

if( st && st->stackIndex > 0 )

{

ret = st->base[st->stackIndex];

st->stackIndex--;

}

return ret;

}

/\* push operation of stack \*/

void push(stack\_t \*st, node\_t \*node)

{

if( st )

{

st->stackIndex++;

st->base[st->stackIndex] = node;

}

}

/\* Iterative insertion

Recursion is least preferred unless we gain something

\*/

node\_t \*insert\_node(node\_t \*root, node\_t\* node)

{

/\* A crawling pointer \*/

node\_t \*pTraverse = root;

node\_t \*currentParent = root;

// Traverse till appropriate node

while(pTraverse)

{

currentParent = pTraverse;

if( node->data < pTraverse->data )

{

/\* left subtree \*/

pTraverse = pTraverse->left;

}

else

{

/\* right subtree \*/

pTraverse = pTraverse->right;

}

}

/\* If the tree is empty, make it as root node \*/

if( !root )

{

root = node;

}

else if( node->data < currentParent->data )

{

/\* Insert on left side \*/

currentParent->left = node;

}

else

{

/\* Insert on right side \*/

currentParent->right = node;

}

return root;

}

/\* Elements are in an array. The function builds binary tree \*/

node\_t\* binary\_search\_tree(node\_t \*root, int keys[], int const size)

{

int iterator;

node\_t \*new\_node = NULL;

for(iterator = 0; iterator < size; iterator++)

{

new\_node = (node\_t \*)malloc( sizeof(node\_t) );

/\* initialize \*/

new\_node->data = keys[iterator];

new\_node->left = NULL;

new\_node->right = NULL;

/\* insert into BST \*/

root = insert\_node(root, new\_node);

}

return root;

}

node\_t \*k\_smallest\_element\_inorder(stack\_t \*stack, node\_t \*root, int k)

{

stack\_t \*st = stack;

node\_t \*pCrawl = root;

/\* move to left extremen (minimum) \*/

while( pCrawl )

{

push(st, pCrawl);

pCrawl = pCrawl->left;

}

/\* pop off stack and process each node \*/

while( pCrawl = pop(st) )

{

/\* each pop operation emits one element

in the order

\*/

if( !--k )

{

/\* loop testing \*/

st->stackIndex = 0;

break;

}

/\* there is right subtree \*/

if( pCrawl->right )

{

/\* push the left subtree of right subtree \*/

pCrawl = pCrawl->right;

while( pCrawl )

{

push(st, pCrawl);

pCrawl = pCrawl->left;

}

/\* pop off stack and repeat \*/

}

}

/\* node having k-th element or NULL node \*/

return pCrawl;

}

/\* Driver program to test above functions \*/

int main(void)

{

node\_t\* root = NULL;

stack\_t stack = { {0}, 0 };

node\_t \*kNode = NULL;

int k = 5;

/\* Creating the tree given in the above diagram \*/

root = binary\_search\_tree(root, ele, ARRAY\_SIZE(ele));

kNode = k\_smallest\_element\_inorder(&stack, root, k);

if( kNode )

{

printf("kth smallest elment for k = %d is %d", k, kNode->data);

}

else

{

printf("There is no such element");

}

getchar();

return 0;

}

**Method 2: Augmented  Tree Data Structure.**

The idea is to maintain rank of each node. We can keep track of elements in a subtree of any node while building the tree. Since we need K-th smallest element, we can maintain number of elements of left subtree in every node.

Assume that the root is having N nodes in its left subtree. If K = N + 1, root is K-th node. If K < N+1, we will continue our search (recursion) for the Kth smallest element in the left subtree of root. If K > N + 1, we continue our search in the right subtree for the (K – N – 1)-th smallest element. Note that we need the count of elements in left subtree only.

Time complexity: O(n) where n is total nodes in tree.

**Algorithm:**

start:

if K = root.leftElement + 1

root node is the K th node.

goto stop

else if K > root.leftElements

K = K - (root.leftElements + 1)

root = root.right

goto start

else

root = root.left

goto srart

stop:

**Implementation:**

#include <stdio.h>

#include <stdlib.h>

#define ARRAY\_SIZE(arr) sizeof(arr)/sizeof(arr[0])

typedef struct node\_t node\_t;

/\* Binary tree node \*/

struct node\_t

{

int data;

int lCount;

node\_t\* left;

node\_t\* right;

};

/\* Iterative insertion

Recursion is least preferred unless we gain something

\*/

node\_t \*insert\_node(node\_t \*root, node\_t\* node)

{

/\* A crawling pointer \*/

node\_t \*pTraverse = root;

node\_t \*currentParent = root;

// Traverse till appropriate node

while(pTraverse)

{

currentParent = pTraverse;

if( node->data < pTraverse->data )

{

/\* We are branching to left subtree

increment node count \*/

pTraverse->lCount++;

/\* left subtree \*/

pTraverse = pTraverse->left;

}

else

{

/\* right subtree \*/

pTraverse = pTraverse->right;

}

}

/\* If the tree is empty, make it as root node \*/

if( !root )

{

root = node;

}

else if( node->data < currentParent->data )

{

/\* Insert on left side \*/

currentParent->left = node;

}

else

{

/\* Insert on right side \*/

currentParent->right = node;

}

return root;

}

/\* Elements are in an array. The function builds binary tree \*/

node\_t\* binary\_search\_tree(node\_t \*root, int keys[], int const size)

{

int iterator;

node\_t \*new\_node = NULL;

for(iterator = 0; iterator < size; iterator++)

{

new\_node = (node\_t \*)malloc( sizeof(node\_t) );

/\* initialize \*/

new\_node->data = keys[iterator];

new\_node->lCount = 0;

new\_node->left = NULL;

new\_node->right = NULL;

/\* insert into BST \*/

root = insert\_node(root, new\_node);

}

return root;

}

int k\_smallest\_element(node\_t \*root, int k)

{

int ret = -1;

if( root )

{

/\* A crawling pointer \*/

node\_t \*pTraverse = root;

/\* Go to k-th smallest \*/

while(pTraverse)

{

if( (pTraverse->lCount + 1) == k )

{

ret = pTraverse->data;

break;

}

else if( k > pTraverse->lCount )

{

/\* There are less nodes on left subtree

Go to right subtree \*/

k = k - (pTraverse->lCount + 1);

pTraverse = pTraverse->right;

}

else

{

/\* The node is on left subtree \*/

pTraverse = pTraverse->left;

}

}

}

return ret;

}

/\* Driver program to test above functions \*/

int main(void)

{

/\* just add elements to test \*/

/\* NOTE: A sorted array results in skewed tree \*/

int ele[] = { 20, 8, 22, 4, 12, 10, 14 };

int i;

node\_t\* root = NULL;

/\* Creating the tree given in the above diagram \*/

root = binary\_search\_tree(root, ele, ARRAY\_SIZE(ele));

/\* It should print the sorted array \*/

for(i = 1; i <= ARRAY\_SIZE(ele); i++)

{

printf("\n kth smallest elment for k = %d is %d",

i, k\_smallest\_element(root, i));

}

getchar();

return 0;

}

Let each node in the BST have a field that number of elements in returns the its left subtree. Let the left subtree of node T contain only elements smaller than T and the right subtree only elements larger than or equal to T.

Now, suppose we are at node T:

1. **k == num\_elements(left subtree of T)+1**, then the answer we're looking for is the value in node T
2. **k > num\_elements(left subtree of T)+1** then obviously we can ignore the left subtree, because those elements will also be smaller than the kth smallest. So, we reduce the problem to finding the k - num\_elements(left subtree of T) -1 smallest element of the right subtree.
3. **k < num\_elements(left subtree of T)**, then the kth smallest is somewhere in the left subtree, so we reduce the problem to finding the kth smallest element in the left subtree.

public int ReturnKthSmallestElement1(int k)

{

Node node = Root;

int count = k;

int sizeOfLeftSubtree = 0;

while(node != null)

{

sizeOfLeftSubtree = node.SizeOfLeftSubtree();

if (sizeOfLeftSubtree + 1 == count)

return node.Value;

else if (sizeOfLeftSubtree < count)

{

node = node.Right;

count -= sizeOfLeftSubtree+1;

}

else

{

node = node.Left;

}

}

return -1;

}

## Given a modified BST where each node carries extra information of the total number of nodes below it. Find the Kth smallest number in O(logn) time.

Node\* Findkth(Node\* root, int k) /\* k > 0, 1st, 2nd, 3rd, 4th, ... kth \*/

{

if (!root || k < 1) return NULL;

if (!root->left)

{

if (k == 1) return root;

else return Findkth(root->right, k-1);

}

else

{

if (k < root->left->countBelow+2)

return Findkth(root->left, k);

else if (k == root->left->countBelow+2)

return root;

else

return Findkth(root->right, k-root->left->countBelow-2);

}

}

## Given a binary tree, convert into a doubly linked list.The list must be as if the tree is traversed in zig-zag order from top to botton. (left to right in one level and right to level in the next)

**Sol1 :With stack**

<http://stackoverflow.com/questions/5282052/convert-binary-tree-in-zigzag-order-to-a-doubly-linked-list>

Make use of two stacks (CurrentStack and NextOrderStack) and a variable that tell us whether to traverse from left-to-right or right-to-left.

1. Start with the root node and place it in the CurrentStack.  
2. While the current stack is not empty:  
3. Pop the value from the current stack, print the value and check the order from the variable.  
(i) if order is left to right, insert the left child of the popped node into the NextOrderStack first and then the right child of the popped node.  
(ii) if order is right to left, insert the right child of the popped node into the NextOrderStack first and then the left child of the popped node.  
4. Change the stack from CurrentStack and NextOrderStack and repeat step 2 and 3.  
5. Continue these till both the stacks become empty.

Use 2 stacks

stack s1,s2

s1.push(root)

while(s1!=null){

while(s1!=null){

s=s1.pop()

add s to list

s2.push(s.left)

s2.push(s.right)

}

while(s2!=null){

s=s2.pop()

add s to list

s1.push(s.right)

s1.push(s.left)

}

}

Ex:

1

2 3

4 5 6 7

8 9

Traversal: 1 3 2 4 5 6 7 9 8

**Sol2: Without stack**

struct node

{

int data;

node \*left;

node \*right;

};

void PrintGivenLevel(struct node \*head,int level,int ltr,int cnt)

{

static struct node \*first=NULL,\*f=NULL,\*last=NULL;

if(cnt==0)

{

f=first;

first=NULL;

last=NULL;

}

if(head==NULL)

return;

if(level==1)

{

if(!first)

{

first=head;

last=head;

first->left=NULL;

}

else

{

last->right=head;

head->left=last;

head->right=NULL;

last=head;

}

}

if(level>0)

{

if(ltr == 1)

{

PrintGivenLevel(head->right,level-1,ltr,1);

PrintGivenLevel(head->left,level-1,ltr,1);

}

else

{

PrintGivenLevel(head->left,level-1,ltr,1);

PrintGivenLevel(head->right,level-1,ltr,1);

}

}

if(f&&(cnt==0))

{

last->right=f;

f->left=last;

}

}

int height(struct node \*head)

{

int lh,rh;

if(head==NULL)

return 0;

lh=height(head->left);

rh=height(head->right);

return lh>=rh?(lh+1):(rh+1);

}

void Print(struct node \*head)

{

int i=0;

int ltr;

int h=height(head);

if(h%2==0)

ltr= -1;

else

ltr= 1;

for(i=h;i>0; --i)

{

PrintGivenLevel(head,i,ltr,0);

ltr= -ltr;

}

}

void PrintLinkedList(node \*head)

{

while(head != NULL) {

printf("%d ", head->data);

head = head->right;

}

printf("\n");

}

int main()

{

node a,b,c,d,e,f;

a.data = 1;

b.data = 2;

c.data = 3;

d.data = 4;

e.data = 5;

f.data = 6;

a.left = &b;

a.right = &c;

b.left = &d;

b.right = &e;

c.right = &f;

c.left = d.left = d.right = e.left = e.right = f.left = f.right = NULL;

Print(&a);

PrintLinkedList(&a);

}

## Given a tree, in addition to the left and right pointer, it has a third pointer, which is set to NULL. Set the third pointer to a node, which will be the successor of the current node, when the tree is traversed in the zig-zag order. In other words, if we traverse the tree using this third pointer alone, then we will be traversing the tree in the zig-zag order.

Input:

(Plz construct the tree using the pre-order and in-order traversals)

Pre-order: 1 2 4 5 3 6 7

In-order: 4 2 5 1 6 3 7

So, after the pointer is fixed, the traversal of the tree using the third pointer should give,

1 3 2 4 5 6 7

## Given a binary tree, print all root-to-leaf paths

/\* Recursive helper function -- given a node, and an array containing

the path from the root node up to but not including this node,

print out all the root-leaf paths.\*/

void printPathsRecur(struct node\* node, int path[], int pathLen)

{

if (node==NULL)

return;

/\* append this node to the path array \*/

path[pathLen] = node->data;

pathLen++;

/\* it's a leaf, so print the path that led to here \*/

if (node->left==NULL && node->right==NULL)

{

printArray(path, pathLen);

}

else

{

/\* otherwise try both subtrees \*/

printPathsRecur(node->left, path, pathLen);

printPathsRecur(node->right, path, pathLen);

}

}

## Given a Tree (not binary Tree), print only leaf nodes with it's path from Root

## Given a binary tree which contains values at each node, find whether the path exist from root to the "LEAF NODE" such that sum of the values of the path nodes is equal to the GIVEN SUM. If so return true or else return false

bool IsSum (root, sum) {

if(root == null)

return false;

if ((sum == 0) && (root->left == null) && (root->right == null)) //reached root and sum is found

return true;

else if ((root->left == null) && (root->right == null)) // reached leaf node but sum not found

return false;

else

return (IsSum(root->left, sum - root->val) ||IsSum(root->right, sum - root->val) )

}

## Given a binary tree, find the path from the root to all nodes whose id is multiples of five.

void Printpaths(struct node \*r,int Path[],int pathlen)

{

if(r==NULL)

return;

Path[pathlen++]=r->value;

if(Path->value%5==0)

{

for(i=0;i<pathlen;i++)

printf("%d ",path[i]);

printf("\n");

}

Printpaths(r->left,Path,pathlen);

Printpaths(r->right,Path,pathlen);

}

## Given a n-ary tree. A random leaf node will be selected. Imagine that you are now holding the tree with your hand from that node. All other nodes will now fall under gravity. Write a function to perform this transformation.

Assumption: We don't need to do any rebalancing on the new tree.

If we hold the tree with the selected leaf node, all the nodes will fall under gravity.  
The new tree will be one obtained by reversing all the pointers starting from the selected leaf node back to the root. The rest of the tree doesn't need any change.

1

2 3

4 5 6 7

8 9

6

3 7

1

2

4 5

8 9

TreeTrasnformUnderGravity(Node\* n, Node\* p)

{

if(n->parent) // check is for root node which doesn't have the parent link

{

n->addson(n->parent); // Add n's parent as n's son

n->parent->removeSon(n); //Reomve the parent link btwn n's parent and n

}

TreeTrasnformUnderGravity(n->parent, n);

n->parent = p;

}

Call -> TreeTrasnformUnderGravity(6, NULL);

## Given a tree, parse the tree using breadth first search and then find the last element in the tree. After finding the last element, replace that element with the root node. Eg. Input:

A

B C

D E F G

Output:

G

B C

D E F A

Sol:

The tricky part of this is finding the parent of the rightmost node. This is easy in a heap structure. For any given element, its parent is going to be index / 2. So create a list<T> of Nodes. The root node is added twice (to make the root node index 1 and not zero to preserve the heap parent / child calculation). Set an enumerator to the List's 1st position. "Pop" the first position by Pushing each child node of the current node onto the list, then advance the enumerator one position. Continue doing so until all nodes are on the list. The last node will be at List.Count -1; Its parent node will be (List.Count - 1) / 2; Examine that parent node to find out if it has a right node. If not, the root node is added to the parent's left side. If it does have a right node, the root node is added to the parent's right side. The last node's left and right pointer are set to the root's left and right. The new rightmost node's left and right pointer are set to null.

## You are given a tree where child nodes point to parent (there is no link from parent to child). Give any two nodes in the tree, write a function to find the lowest common ancestor. Node\* LCA(Node \*p, Node \*q){ …}

The problem is same as finding intersection of two linked lists.

1. Find depth of P & Q p,q

p = P -> ......> Root

q = Q -> .... > Root

1. diff = p-q;
2. traverse abs(diff) on the longer list.
3. Now both list are same length. Traverse each node on both the lists till common list is reached.

## Given two nodes of tree .Find their first common ancestor

**If the tree is BST**

Sol1: Iterative

LCA of BST: <http://www.geeksforgeeks.org/archives/1029>

Assumption: val1 is less than val2

int LCA(node\* root,int val1,int val2)

{

if(!root) return 0;

node\* curr = root;

node\* par = root;

while(curr)

{

if(curr->data < val1 && curr->data < val2)

{

par = curr;

curr = curr->right;

}

else if(curr->data>val1 && curr->data>val2)

{

par = curr;

curr = curr->left;

}

else if(val1==curr->data || val2== curr->data)

{

curr = par;

cout<<"\nLowest Common Ancestor of "<<val1<<" and "<<val2<<" is "<<curr->data;

break;

}

else if(curr->data > val1 && curr->data < val2)

{

cout<<curr->data;

break;

}

}

}

**Sol2: Recursive**

public static Node lca(Node n, int x, int y){

if(n==null)

return null;

if(n.value == x || n.value == y){

return n;

}

if(n.value>x && n.value<y)

return n;

Node lca = null;

if(n.value>y){

lca = lca(n.left, x, y);

}else{

lca = lca(n.right, x, y);

}

return lca;

}

**Lowest Common Ancestor of a Binary Tree**

<http://www.ihas1337code.com/2011/07/lowest-common-ancestor-of-a-binary-tree-part-i.html>

**Sol1 : A Top-Down Approach (Worst case O(n2) ):**

First, if the current node is one of the two nodes, it must be the LCA of the two nodes. If not, we count the number of nodes that matches either p or q in the left subtree (which we call totalMatches). If totalMatches equals 1, then we know the right subtree will contain the other node. Therefore, the current node must be the LCA. If totalMatches equals 2, we know that both nodes are contained in the left subtree, so we traverse to its left child. Similar with the case where totalMatches equals 0 where we traverse to its right child.

// Return #nodes that matches P or Q in the subtree.

int countMatchesPQ(Node \*root, Node \*p, Node \*q) {

if (!root) return 0;

int matches = countMatchesPQ(root->left, p, q) + countMatchesPQ(root->right, p, q);

if (root == p || root == q)

return 1 + matches;

else

return matches;

}

Node \*LCA(Node \*root, Node \*p, Node \*q) {

if (!root || !p || !q) return NULL;

if (root == p || root == q) return root;

int totalMatches = countMatchesPQ(root->left, p, q);

if (totalMatches == 1)

return root;

else if (totalMatches == 2)

return LCA(root->left, p, q);

else /\* totalMatches == 0 \*/

return LCA(root->right, p, q);

}

**Sol2 : A Bottom-up Approach (Worst case O(n) ):**

We traverse from the bottom, and once we reach a node which matches one of the two nodes, we pass it up to its parent. The parent would then test its left and right subtree if each contain one of the two nodes. If yes, then the parent must be the LCA and we pass its parent up to the root. If not, we pass the lower node which contains either one of the two nodes (if the left or right subtree contains either p or q), or NULL (if both the left and right subtree does not contain either p or q) up.

Node \*LCA(Node \*root, Node \*p, Node \*q) {

if (!root) return NULL;

if (root == p || root == q) return root;

Node \*L = LCA(root->left, p, q);

Node \*R = LCA(root->right, p, q);

if (L && R) return root; // if p and q are on both sides

return L ? L : R; // either one of p,q is on one side OR p,q is not in L&R subtrees

}

<http://www.ihas1337code.com/2011/07/lowest-common-ancestor-of-a-binary-tree-part-ii.html>

As we trace the two paths from both nodes up to the root, eventually it will merge into one single path. The LCA is the exact first intersection node where both paths merged into a single path. An easy solution is to use a hash table which records visited nodes as we trace both paths up to the root. Once we reached the first node which is already marked as visited, we immediately return that node as the LCA.

Node \*LCA(Node \*root, Node \*p, Node \*q) {

hash\_set<Node \*> visited;

while (p || q) {

if (p) {

if (!visited.insert(p).second)

return p; // insert p failed (p exists in the table)

p = p->parent;

}

if (q) {

if (!visited.insert(p).second)

return q; // insert q failed (q exists in the table)

q = q->parent;

}

}

return NULL;

}

A little creativity is needed here. Since we have the parent pointer, we could easily get the distance (height) of both nodes from the root. Once we knew both heights, we could subtract from one another and get the height’s difference (dh). If you observe carefully from the previous solution, the node which is closer to the root is always dh steps ahead of the deeper node. We could eliminate the need of marking visited nodes altogether. Why?

The reason is simple, if we advance the deeper node dh steps above, both nodes would be at the same depth. Then, we advance both nodes one level at a time. They would then eventually intersect at one node, which is the LCA of both nodes. If not, one of the node would eventually reach NULL (root’s parent), which we conclude that both nodes are not in the same tree. However, that part of code shouldn’t be reached, since the problem statement assumed that both nodes are in the same tree.

int getHeight(Node \*p) {

int height = 0;

while (p) {

height++;

p = p->parent;

}

return height;

}

// As root->parent is NULL, we don't need to pass root in.

Node \*LCA(Node \*p, Node \*q) {

int h1 = getHeight(p);

int h2 = getHeight(q);

// swap both nodes in case p is deeper than q.

if (h1 > h2) {

swap(h1, h2);

swap(p, q);

}

// invariant: h1 <= h2.

int dh = h2 - h1;

for (int h = 0; h < dh; h++)

q = q->parent;

while (p && q) {

if (p == q) return p;

p = p->parent;

q = q->parent;

}

return NULL; // p and q are not in the same tree

}

## Given a BST and two values m and n . We need to find out all the nodes whose values are in range of m and n .

printNodes(Node root,int m,int n)

{

if(root==null)

return;

if(root.data>=m&&root.data<=n)

print(root.data);

if(m<root.data)

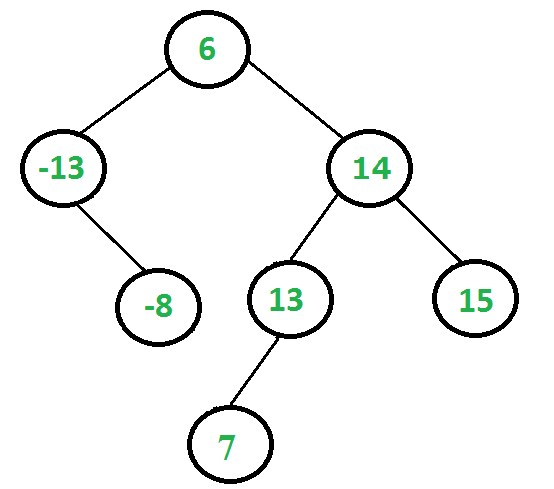
printNode(root.left,m,n);

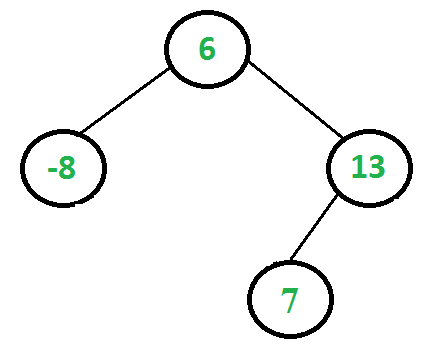
if(n>root.data)

printNode(root.right,m,n);

}

## Remove BST keys outside the given range

Given a Binary Search Tree (BST) and a range [min, max], remove all keys which are outside the given range. The modified tree should also be BST. For example, consider the following BST and range [-10, 13].   
[](http://www.geeksforgeeks.org/wp-content/uploads/BinaryTree14.png)

The given tree should be changed to following. Note that all keys outside the range [-10, 13] are removed and modified tree is BST.  
[](http://www.geeksforgeeks.org/wp-content/uploads/BinaryTreeModified2.png)

There are two possible cases for every node.  
**1)** Node’s key is outside the given range. This case has two sub-cases.  
…….**a)** Node’s key is smaller than the min value.  
…….**b)** Node’s key is greater that the max value.  
**2)** Node’s key is in range.

We don’t need to do anything for case 2. In case 1, we need to remove the node and change root of sub-tree rooted with this node.  
The idea is to fix the tree in Postorder fashion. When we visit a node, we make sure that its left and right sub-trees are already fixed. In case 1.a), we simply remove root and return right sub-tree as new root. In case 1.b), we remove root and return left sub-tree as new root.

Following is C++ implementation of the above approach.

// A C++ program to remove BST keys outside the given range

#include<stdio.h>

#include <iostream>

using namespace std;

// A BST node has key, and left and right pointers

struct node

{

int key;

struct node \*left;

struct node \*right;

};

// Resmoves all nodes having value outside the given range and returns the root

// of modified tree

node\* removeOutsideRange(node \*root, int min, int max)

{

// Base Case

if (root == NULL)

return NULL;

// First fix the left and right subtrees of root

root->left = removeOutsideRange(root->left, min, max);

root->right = removeOutsideRange(root->right, min, max);

// Now fix the root. There are 2 possible cases for toot

// 1.a) Root's key is smaller than min value (root is not in range)

if (root->key < min)

{

node \*rChild = root->right;

delete root;

return rChild;

}

// 1.b) Root's key is greater than max value (root is not in range)

if (root->key > max)

{

node \*lChild = root->left;

delete root;

return lChild;

}

// 2. Root is in range

return root;

}

// A utility function to create a new BST node with key as given num

node\* newNode(int num)

{

node\* temp = new node;

temp->key = num;

temp->left = temp->right = NULL;

return temp;

}

// A utility function to insert a given key to BST

node\* insert(node\* root, int key)

{

if (root == NULL)

return newNode(key);

if (root->key > key)

root->left = insert(root->left, key);

else

root->right = insert(root->right, key);

return root;

}

// Utility function to traverse the binary tree after conversion

void inorderTraversal(node\* root)

{

if (root)

{

inorderTraversal( root->left );

cout << root->key << " ";

inorderTraversal( root->right );

}

}

// Driver program to test above functions

int main()

{

node\* root = NULL;

root = insert(root, 6);

root = insert(root, -13);

root = insert(root, 14);

root = insert(root, -8);

root = insert(root, 15);

root = insert(root, 13);

root = insert(root, 7);

cout << "Inorder traversal of the given tree is: ";

inorderTraversal(root);

removeOutsideRange(root, -10, 13);

cout << "\nInorder traversal of the modified tree is: ";

inorderTraversal(root);

return 0;

}

Output:

Inorder traversal of the given tree is: -13 -8 6 7 13 14 15

Inorder traversal of the modified tree is: -8 6 7 13

**Time Complexity:** O(n) where n is the number of nodes in given BST.

## Given a binary tree, write a code to update each node with the sum of values of its subtree.

int UpdateTree(Node\* n) {

if(n==NULL)

return 0;

n->data += UpdateTree(n->left)+UpdateTree(n->right);

return n->data;

}

## Given a Binary Tree ( Comprising of +ve & -ve numbers ), represent each node of the tree with sum of its LeftSubTree & RightSubTree

For Example :-

10 | 20

-2 6 | 4 12

8 -4 7 5 | 8 -4 7 5

just do the postorder traversal of tree and in each recursive call check weather the parent have same valeue as the sum of its childrean else just do maintain it :)

void BinaryTreeSum(struct node \*T)

{

if(!T||T->left==NULL&&T->right==NULL)

return;

BinaryTreeSum(T->left);

BinaryTreeSum(T->right);

if(T->left&&T->right)

sum=T->left->data+T->right->data;

else if(T->left&&!T->right)

sum=T->left->data;

else

sum=T-right->data;

T->data=sum;

}

## Given a binary search tree and a value X, find the node with value immediately greater than X.

int nextgreater(int x, struct node \*r)

{

struct node \*temp=r;

inorderSucc=NULL;

if(!r)

{

printf("not exist");

return -1;

}

while(temp != NULL)

{

if(temp->value==x)

break;

if(x > temp->value)

temp=temp->right;

else //if(temp->value>x)

{

inorderSucc=temp;

temp=temp->left;

}

}

if(temp==NULL)

{

printf("NOT exist");

return -1;

}

if(temp->right)

{

temp=temp->right;

while(temp->left)

temp=temp->left;

}

else if(inorderSucc)

{

return inorderSucc;

}

else

return -1;

return temp;

}

## Find the maximum subtree in a given binary tree which is a binary search tree

**Java Sol**

public class MaxBSTinBTree

{

public static MaxBSTHelper GetMaxBSToutOfBTree(BTreeNode node)

{

MaxBSTHelper retNode = new MaxBSTHelper();

if (node.LeftNode == null && node.RightNode == null)

{

retNode.Node = node;

retNode.Size = 1;

retNode.IsBst = true;

}

else

{

MaxBSTHelper left = null;

MaxBSTHelper right = null;

if (node.LeftNode != null)

left = GetMaxBSToutOfBTree(node.LeftNode);

if (node.RightNode != null)

right = GetMaxBSToutOfBTree(node.RightNode);

if (left == null && right != null)

{

if ((int)node.Data <= (int)right.Node.Data && right.IsBst)

{

retNode.Node = node;

retNode.Size = 1 + right.Size;

retNode.IsBst = true;

}

else

{

retNode = right;

}

}

else if (right == null && left != null)

{

if ((int)node.Data >= (int)left.Node.Data && left.IsBst)

{

retNode.Node = node;

retNode.Size = 1 + left.Size;

retNode.IsBst = true;

}

else

{

retNode = left;

}

}

else if (right != null && left != null)

{

if ((int)node.Data >= (int)left.Node.Data && left.IsBst && (int)node.Data <= (int)right.Node.Data && right.IsBst)

{

retNode.Node = node;

retNode.Size = 1 + left.Size + right.Size;

retNode.IsBst = true;

}

else

{

if (left.Size>right.Size)

retNode = left;

else

retNode = right;

}

}

}

return retNode;

}

#region inner class

public class MaxBSTHelper

{

private BTreeNode m\_node;

public BTreeNode Node

{

get { return m\_node; }

set { m\_node = value; }

}

private int m\_size;

public int Size

{

get { return m\_size; }

set { m\_size = value; }

}

private bool m\_isBst;

public bool IsBst

{

get { return m\_isBst; }

set { m\_isBst = value; }

}

}

#endregion inner class

}

**C++ sol**

int larg\_bst(struct node \*n,struct node \*\*subroot,bool \*flag)

{

static int hp=0,hc=0;

bool flag\_l,flag\_r;

int hl,hr;

if(n==NULL)

return 0;

hl=larg\_bst(n->l,subroot,&flag\_l);

hr=larg\_bst(n->r,subroot,&flag\_r);

if(n->l)

{

if(n->r)

{

if(n->v > n->l->v && n->v < n->r->v)

\*flag=true;//if left ,right both exist and are bst then pass true to parent

}

else

{

if(n->v > n->l->v)

\*flag=true;flag\_r=true;

}

}

else if(n->r)

{

if(n->v < n->r->v)

\*flag=true;flag\_l=true;

}

else

{

flag\_l=true;flag\_r=true;\*flag=true; //if no child there then bst and pass true to parent

}

//now for each node checking the statuc of left and right flag and setting subroot if it is keeping in mind the height

if(flag\_l==true && flag\_r==true)

{

hc=hl>hr?hl:hr;

hc++;

if(hc > hp)

{

hp=hc;

\*subroot=n;

// printf("found new root:%d h:%d %d\n",(\*subroot)->v,hc,n->v);

}

}

return hc;

}

## If a binary tree is a binary search tree.

**Sol1: Recursive1 MinValue and max value is assumed to be implemented.**

/\*

Returns true if a binary tree is a binary search tree.

\*/

int isBST(struct node\* node) {

if (node==NULL) return(true);

// false if the max of the left is > than us

// (bug -- an earlier version had min/max backwards here)

if (node->left!=NULL && maxValue(node->left) > node->data)

return(false);

// false if the min of the right is <= than us

if (node->right!=NULL && minValue(node->right) <= node->data)

return(false);

// false if, recursively, the left or right is not a BST

if (!isBST(node->left) || !isBST(node->right))

return(false);

// passing all that, it's a BST

return(true);

}

/\*

Given a non-empty binary search tree,

return the minimum data value found in that tree.

Note that the entire tree does not need to be searched.

\*/

int minValue(struct node\* node) {

struct node\* current = node;

// loop down to find the leftmost leaf

while (current->left != NULL) {

current = current->left;

}

return(current->data);

}

**Sol2: Recursive2**

int isBSTUtil(struct node\* node, int min, int max);

/\* Returns true if the given tree is a binary search tree

(efficient version). \*/

int isBST(struct node\* node)

{

return(isBSTUtil(node, INT\_MIN, INT\_MAX));

}

/\* Returns true if the given tree is a BST and its

values are >= min and <= max. \*/

int isBSTUtil(struct node\* node, int min, int max)

{

/\* an empty tree is BST \*/

if (node==NULL)

return 1;

/\* false if this node violates the min/max constraint \*/

if (node->data < min || node->data > max)

return 0;

/\* otherwise check the subtrees recursively,

tightening the min or max constraint \*/

return

isBSTUtil(node->left, min, node->data) &&

isBSTUtil(node->right, node->data+1, max);

}

## Write the code to test if a given binary tree is balanced or not? Write/give test cases

bool isBalanced = true;

int isTreeBalanced(int level, node\* ptr)

{

if(ptr == null) return level;

int leftDepth = isTreeBalanced(level+1, ptr->left);

int rightDepth = isTreeBalanced(level+1, ptr->right);

if(abs(leftDepth - rightDepth) > 1)

isBalanced = false;

}

## To get mirror image of a binary tree.

Sol1:

To get the mirror image of the binary tree we have to just traverse the tree in postorder and in a bottom up fashion we have to swap the right son to left

code :

void MirrorImage(struct node \*R)

{

struct node \*temp=NULL;

if(R==NULL||(R->left==NULL &&R->right==NULL))

return;

MirrorImage(R->left);

MirrorImage(R->right);

temp=R->left;

R->left=R->right;

R->right=temp;

}

Sol2: Mirror binary tree.

<http://interviewcracker.wordpress.com/2008/06/13/mirror-binary-tree/>

typedef struct block

{

char data;

struct block \*left;

struct block \*right;

}node;

void mirror(node \*start)

{

if(start == NULL) return;

node \*temp;

temp = start->left;

start->left = start->right;

start->right = temp;

mirror(start->left);

mirror(start->right);

}

void MirrorBinaryTreeIterative(node \*start)

{

node \*stack[50] = {0};

int top = 0;

// adding root to stack

stack[top++] = start;

// while stack is not empty

while(top != 0)

{

top = top-1;

if(stack[top] != NULL)

{

node \*curNode = stack[top];

// node swaping

node \*temp;

temp = curNode->left;

curNode->left = curNode->right;

curNode->right = temp;

// adding left and right nodes to the stack

stack[top++] = curNode->left;

stack[top++] = curNode->right;

}

}

}

## Convert Sorted Doubly Linked List into a Binary Tree

You must use the following algorithm to build the search tree:   
1. Find the middle node, build a root node (TNode) with an info field the same as the middle node   
2. Unlink the middle node from the list (so the next field of the node before it points to NULL, and the prev field of the node after it points to NULL) --- this step is necessary for the next step (so that MiddleNode will work) --- you may want to store pointers to the nodes before and after the middle node.   
3. Make left and right subtrees recursively, using the list before the middle node for the left subtree and the list after the middle node for the right subtree.   
4. Link the middle node back into the list (which is why you may have stored pointers in step 2).

struct node

{

int data;

node\* prev; // or left;

node\* next; // or right;

};

node \*totree(node \*r)

{

if(r)

{

node\* x=r;

node\* y=r;

while(y)

{

x++; y+=2;

}

x->prev->next=NULL;

x->prev = totree(r);

x->next = totree(x->next);

}

return r;

}

## WAP to check a tree is symmetric.

bool isSymmetric(Node \*r1, Node \*r2)

{

if (r1 == NULL && r2 == NULL)

return true;

if (r1 == NULL || r2 == NULL)

return false;

return (r1->data == r2->data && isSymmetric(r1->left, r2->right) && isSymmetric(r1->right, r2->left));

}

## Find median of BST.

So, a simple algo for median in BST would be:  
1) Use any algo to count the number of nodes in the BST. Let it be n.  
2) Use morris inorder(no recursion/no stacks-all constraint met ) to traverse the tree. For each node visited increment the counter.   
a) If n is even then return avg(n/2,n/2+1)(counter == n/2)  
b) If n is odd return when counter == (n+1)/2(1-based indexing)

<http://www.geeksforgeeks.org/archives/6358>

Modify below algo for keeping a counter..

void MorrisTraversal(struct tNode \*root)

{

struct tNode \*current,\*pre;

if(root == NULL)

return;

current = root;

while(current != NULL)

{

if(current->left == NULL)

{

printf(" %d ", current->data);

current = current->right;

}

else

{

/\* Find the inorder predecessor of current \*/

pre = current->left;

while(pre->right != NULL && pre->right != current)

pre = pre->right;

/\* Make current as right child of its inorder predecessor \*/

if(pre->right == NULL)

{

pre->right = current;

current = current->left;

}

/\* Revert the changes made in if part to restore the original

tree i.e., fix the right child of predecssor \*/

else

{

pre->right = NULL;

printf(" %d ",current->data);

current = current->right;

} /\* End of if condition pre->right == NULL \*/

} /\* End of if condition current->left == NULL\*/

} /\* End of while \*/

}

## Given an n-ary tree of resources arranged hierarchically. A process needs to lock a resource node in order to use it. But a node cannot be locked if any of its descendant or ancestor is locked. You are supposed to: -> write the structure of node -> write codes for

Islock()- returns true if a given node is locked and false if it is not  
Lock()- locks the given node if possible and updates lock information  
Unlock()- unlocks the node and updates information.  
Codes should be :

Islock –O(1)  
Lock()- O(log n)  
unLock()- O(log n)

class Tree {

private:

// Tree Structure related.

List <TreeNode \*> \_children;

TreeNode \* \_parent;

// Locking related.

bool \_locked = false;

uint \_numDescendantsLocked = 0;

};

bool Tree::IsLocked() {

return \_locked;

}

bool Tree::Lock() {

// Any descendants locked?

if (\_numDescendantsLocked > 0) { return false;}

// Check if any ancestor is locked.

Tree \*parent = \_parent;

while (parent) {

if (parent->IsLocked()) {

return false;

}

parent = parent->\_parent;

}

// Can lock now.

parent = \_parent;

while(parent) {

parent->\_numDescendantsLocked++;

parent = parent->\_parent;

}

\_locked = true;

return true;

}

## Find diameter of a binary tree in O(n)

<http://www.geeksforgeeks.org/archives/5687>

**Sol1:**

The diameter of a tree (sometimes called the width) is the number of nodes on the longest path between two leaves in the tree

/\*The second parameter is to store the height of tree.

Initially, we need to pass a pointer to a location with value

as 0. So, function should be used as follows:

int height = 0;

struct node \*root = SomeFunctionToMakeTree();

int diameter = diameterOpt(root, &height); \*/

int diameterOpt(struct node \*root, int\* height)

{

/\* lh --> Height of left subtree

rh --> Height of right subtree \*/

int lh = 0, rh = 0;

/\* ldiameter --> diameter of left subtree

rdiameter --> Diameter of right subtree \*/

int ldiameter = 0, rdiameter = 0;

if(root == NULL)

{

\*height = 0;

return 0; /\* diameter is also 0 \*/

}

/\* Get the heights of left and right subtrees in lh and rh

And store the returned values in ldiameter and ldiameter \*/

ldiameter = diameterOpt(root->left, &lh);

rdiameter = diameterOpt(root->right, &rh);

/\* Height of current node is max of heights of left and

right subtrees plus 1\*/

\*height = max(lh, rh) + 1;

return max(lh + rh + 1, max(ldiameter, rdiameter));

}

**Other sol:**

int diameter(Node \*root, int \*height)

{

if (root==NULL)

{

\*height=0;

return 0

}

int ld, rd, lh, rh;

ld=rd=lh=rh=0;

ld = diameter(root->left, &lh);

rd = diameter(root->right, &rh);

\*height=max(lh,rh)+1;

return max(ld, max(rd, lh+rh+1));

}

## Given a binary tree. Find the minimum element at a given depth.

Sol1:

breadth first traversal

Sol2 : Depth first traversal Maintain depth and current\_min at required depth.

FindMinAtDepth(Tree \*t, int depth) {

return FindMinAtDepthInternal(t, depth, MAX\_INT);

}

FindMinAtDepthInternal(Tree \*t, int depth, int cur\_min) {

if (!t) return cur\_min;

// Found at node at require depth. Compare with min seen so far and

// return the right value

if (depth == 0) return t->value < cur\_min ? t->value : cur\_min;

// Need to go deeper.

int left\_min = FindMinAtDepthInternal(t->left, depth-1, cur\_min);

return FindMinAtDepthInternal(t->right, depth-1, left\_min);

}

public static Node findMinByLevel(Node n, int level){

if(n==null || level==0){

return n;

}

Node left = findMinByLevel(n.left, level-1);

Node right = findMinByLevel(n.right, level-1);

if(left==null && right == null){

return null;

}

if(left==null && right != null){

return right;

}

if(left!=null && right == null){

return left;

}

if(left.value<right.value)

return left;

else

return right;

}

## Vertical Sum of a Binary Tree OR Vertical split of a binary tree

#include<iostream>

#include<map>

using namespace std;

struct tree

{

int data;

struct tree\* left;

struct tree\* right;

};

typedef struct tree Tree;

Tree\* Node(int d)

{

Tree\* T=(Tree\*)malloc(sizeof(Tree));

T->data=d;

T->left = T->right = NULL;

return T;

}

map<int,int>verticalSum;

void get\_Vertical\_Sum(Tree\* T,int i)

{

if(T==NULL)

return;

else

{

verticalSum[i]+=T->data; // If split is needed we will add T to the list at index i.

get\_Vertical\_Sum(T->left,i-1);

get\_Vertical\_Sum(T->right,i+1);

}

}

int main()

{

//Construct the Tree

/\*

1

2 3

4 5 6 7

\*/

Tree\* T=NULL;

T = Node(1);

T->left = Node(2);

T->right = Node(3);

T->left->left = Node(4);

T->left->right = Node(5);

T->right->left = Node(6);

T->right->right = Node(7);

verticalSum.clear();

get\_Vertical\_Sum(T,0);

for(map<int,int>::iterator it=verticalSum.begin();it!=verticalSum.end();it++)

{

printf("%d ",(\*it).second);

}

printf("\n");

getchar();

return 0;

}

## Given a Node in a binary tree . We need to find out all the nodes at K distance from that node .

## For the given number need to find out the possible BST. eg, if the given number is n means we should find BSTs using 1,2..n. n=5 means figure it out using 1,2,3,4,5, how many BST we can make?

Not sure:

Catalan number

(2**n** *C* **n**) / (n+1) = 2n!/n!n!(n+1)

## There is a Directory in which there are subdirectories and recursively have subdirectories. We need to find out is any subdirectory is pointing to its any of its ancestor.

## Given a no. find all pairs of elements in a balanced bst, which sum to this number. Make sure you don’t return same pair again.

Simlar one is above

## construct double linklist out of a BST

Look for similar one

## Print the nodes of binary tree in on sorted order

## Two BST were given and find the largest common bst that exists between these two

Look for similar ques.

## Print a tree layer by layer

## Flatten BST to sorted list | Decreasing order

Given a binary search tree, the task is to flatten it to a sorted list in decreasing order. Precisely, the value of each node must be greater than the values of all the nodes at its right, and its left node must be NULL after flattening. We must do it in O(H) extra space where ‘H’ is the height of BST.

**Impoprtant notes:**

1. The reverse inorder traversal gives the output in sorted order too.
2. In order to maintain the traversal the prev node is needed so we can link the prev to the current node.
3. In my code we just keep on inserting the node to front of linked list.

**Mistakes made:**

1. While inserting assign the pointer correctly. n->left = ll and not n->left = ll->left.

// Node of the binary tree

struct node {

    int data;

    node\* left;

    node\* right;

    node(int data)

    {

        this->data = data;

        left = NULL;

        right = NULL;

    }

};

// Function to print flattened

// binary tree

void print(node\* parent)

{

    node\* curr = parent;

    while (curr != NULL)

        cout << curr->data << " ", curr = curr->left;

}

// Function to perform reverse in-order traversal

**void revInorder(node\* curr, node\*& prev)**

**{**

**// Base case**

**if (curr == NULL)**

**return;**

**revInorder(curr->right, prev);**

**prev->left = NULL;**

**prev->right = curr;**

**prev = curr;**

**revInorder(curr->left, prev);**

**}**

// Function to flatten binary tree using

// level order traversal

node\* flatten(node\* parent)

{

    // Dummy node

    node\* dummy = new node(-1);

    // Pointer to previous element

    node\* prev = dummy;

    // Calling in-order traversal

    revInorder(parent, prev);

    prev->left = NULL;

    prev->right = NULL;

    node\* ret = dummy->right;

    // Delete dummy node

    delete dummy;

    return ret;

}

// Driver code

int main()

{

    node\* root = new node(5);

    root->left = new node(3);

    root->right = new node(7);

    root->left->left = new node(2);

    root->left->right = new node(4);

    root->right->left = new node(6);

    root->right->right = new node(8);

    // Calling required function

    print(flatten(root));

    return 0;

}

//My code: It uses the left pointer instead of right pointer as was asked in the questions but its great 😊

void FlatList(node\*& root)

{

    node\* ll = NULL;

    FlatListInternal(root, ll);

    root = ll;

}

void FlatListInternal(node\* root, node\* &ll)

{

  if(!root)

    return;

  FlatListInternal(root->left, ll);

  InsertNode(root, ll);

  FlatListInternal(root->right, ll);

}

void InsertNode(node\* n, node\* &ll)

{

  if(!ll)

  {

    ll = n;

    return ;

  }

  n->left = ll;

  ll = n;

}

# Graphs

## Find the Town Judge

<https://leetcode.com/problems/find-the-town-judge/>

#include <vector>

#include <iostream>

using namespace std;

class Solution {

public:

    int findJudge(const int N, vector<vector<int>>& trust) {

        vector<vector<int>> A(N+1);

        for(int i = 0 ; i < A.size() ; i++)

            A[i] = vector<int>(N+1);

        for(int i = 0; i < trust.size();i++)

        {

            int r = trust[i][0];

            int c = trust[i][1];

            A[r][c] = 1;

        }

        vector<int> v\_cand;

        for(int c = 1; c<=N ; c++)

        {

            v\_cand.push\_back(c);

            for(int r = 1; r<=N; r++)

            {

                if( (r!=c || A[r][c]==1) && A[r][c] != 1)

                {

                    v\_cand.pop\_back();

                    break;

                }

            }

        }

        for( auto cand = v\_cand.begin(); cand != v\_cand.end();)

        {

            bool erased = false;

            for (int c=1; c <= N; c++)

            {

                if(A[\*cand][c] != 0)

                {

                    erased = true;

                    cand = v\_cand.erase(cand);

                    break;

                }

            }

            if(!erased) cand++;

        }

        if(v\_cand.size() == 1)

            return v\_cand[0];

        else

            return -1;

    }

};

int main()

{

    Solution s;

    vector<vector<int>> v = {{1,3},{2,3},{3,1}};

    cout<< s.findJudge(3,v) << endl;

    return 0;

}

Better solution <https://leetcode.com/problems/find-the-town-judge/discuss/242938/JavaC%2B%2BPython-Directed-Graph>

## LC1042. Flower Planting With No Adjacent

<https://leetcode.com/problems/flower-planting-with-no-adjacent/>

<https://leetcode.com/problems/flower-planting-with-no-adjacent/discuss/453470/c%2B%2BEasy-to-understand>

class Solution {

public:

    vector<int> gardenNoAdj(int N, vector<vector<int>>& paths) {

        vector<int> colors(N,-1);

        map<int,list<int> > adjlist;

        for(auto i:paths)

        {

            int u = \*(i.begin());

            int v = \*(i.begin()+1);

            adjlist[u].push\_back(v); 🡨 this is valid, will create a key.

            adjlist[v].push\_back(u);

        }

        for(int i=1;i<=N;i++)

        {

            int node = i;

            bool found=true;

            for(int f=1;f<=4;f++)

            {

                bool found=true;

                for(auto neighbour:adjlist[node])

                {

                    if(colors[neighbour-1]==f)

                    {

                        found=false;

                        break;

                    }

                }

                if(found)

                {

                    colors[node-1]=f;

                    break;

                }

            }

        }

       return colors;

    }

};

# Algorithms

## Types

Dynamic

Greedy

Divide and Conquer

Sorting

Searching

Pattern matching

Graphs

## you have an array of integers, find the longest subarray which consists of numbers that can be arranged in a sequence, e.g.: a = {4,5,1,5,7,4,3,6,3,1,9} max subarray = {5,7,4,3,6}

<http://www.careercup.com/question?id=11256218>

<http://www.careercup.com/question?id=9783960>

## You have given a file containing sentences. Now you are given a sequence of characters. You have to find the starting location of each word containing one of the permutations of the word.

e.g File - 'She submitted her assignment.' Input Sequence - imt. Since the file contains the word 'submitted' containing the sequence 'mit' which is a permutation of 'imt' So it will return 1. Similiarly it will return for all other words.

Sol: use modified version of Rabin-Karp algorithm to solve this. Basically if your hashing function is incremental and sorts the input (for small strings this is really O(1)) before calculating the hash then basically this problem reduces to just scanning the document and matching the hash.

## We have a job sequence like this.

A

/ \

B --C

This case A's dependent jobs are B & C

B's dependent job is C

C's Dependent job is null

If we need to complete any of the steps we should need to complete the dependent object.

This case job sequence will be C->B->A

Can anyone comeup with simple o(n) solution for this?

Solution: We reverse all the edges in the dependency graph and then perform topological sort.

## Edit Distance ?

## In a plane we are given latitude, longitude coordinate, and we are also given a point (having lat, long value). We need to find out the nearest point, in most efficient way.

<http://en.wikipedia.org/wiki/Closest_pair_of_points_problem>

## Given a large file containing 10 bit integers, and a limited memory, sort the entire file and store back to file. Constraints: You cannot read entire file in main memory at a time as memory is limited (small)

Follow up questions: Quick sort working, complexity best,avg,worst

Hint :Use property of 10 bit integer

## How will you find n most frequently occurring patterns in a text file. What data structures would you use? Here, a pattern is not a single word but rather a sequence of words. For instance, "this is a" could be a frequently occurring pattern in the file.

Followup questions:

* What if the file is very large (in GBs)?
* What if the file contains text in multiple languages (english, japanese etc)?

PS: I understand that the most frequently occurring words can be found relatively easily using a hash table or BST. Just can't think of how we can find multi-word strings.

Sol: Suffix trees.

TBD

## you have a Log file that stores the "users id", "time"(irrelevant), and "webpage visited." Find the most common three-sequence webpage visited by the users. For example, you have a log file

joe, 12:30pm,home

joe, 12:31pm,about

joe, 12:32pm,career

james, 12:35pm, home

james, 12:36pm, cart

james, 12:37pm, maps

mary, 12:41pm, home

mary, 12:42pm, about

mary, 12:43pm, career

thus the most common sequence is home->about->career because the sequence occurs twice as oppose to home->car->maps which only occurs once.

In addition, this file is a huge file with many more entries.

Sol 1

I think problem can effectively solved by having a tree whose node represents the page and number of times it has been visited in the sequence from root the given node. Every time a new user browse through the pages, the information is update to this tree.

Let me above eg to demonstrate:

joe, 12:30pm,home

james, 12:35pm, home

joe, 12:31pm,about

joe, 12:32pm,career

james, 12:36pm, cart

joe, 12:40pm, cart

mary, 12:41pm, home

james, 12:37pm, maps

mary, 12:42pm, about

mary, 12:43pm, career

joe, 1:00pm, maps

Initially my tree is empty, Now assume Joe is the first visiter to the site

From this we can deduce that Joe browses the pages in sequence (Home->About->Career->Cart->Map)

So my tree would be

Home (1)->About(1)->Career(1)->Cart(1)->Maps(1)

James is the second user (Home->Cart->Maps), so tree would have few more node

Home(2)->About(1)->Career(1)->Cart(1)->Maps(1)

\

\->Cart(1)->Maps(1)

When mary cames in and browse pages as (Home->About->Career)

Home(3)->About(2)->Career(2)->Cart(1)->Maps(1)

\

\->Cart(1)->Maps(1)

Sol 2

Maybe using graph.

Each webpage is a node. If user visit A and then B, then there is a edge from A to B. Also, we can store a count from A to B. Then we can scan the log and build the graph. After the graph is constructed, do a BFS with 3 degree to find the max count.

## Jump Game: Given an array start from the first element and reach the last by jumping. The jump length can be at most the value at the current position in the array. Optimum result is when u reach the goal in minimum number of jumps.

For ex:

Given array A = {2,3,1,1,4}

possible ways to reach the end (index list)

i) 0,2,3,4 (jump 2 to index 2, then jump 1 to index 3 then 1 to index 4)

ii) 0,1,4 (jump 1 to index 1, then jump 3 to index 4)

Since second solution has only 2 jumps it is the optimum result.

/\*

Algo:

Lets define a function called hop which is number of steps required to reach the end of array from the given position, So

hop(i) = INT\_MAX if a[i] == 0

hop(i) = 1 + min(hop(j) where j is from i + 1 to i + a[i]

hop(0) is the required answer

\*/

#include <stdio.h>

#include <limits.h>

int minhops(int a[], int n) {

int hop[n] ;

int i = n;

while(i--) {

if(a[i] == 0)

hop[i] = INT\_MAX;

else {

if(i + a[i] >= n)

hop[i] = 1;

else {

int j, min = INT\_MAX;

for(j = i + a[i]; j > i; --j){

if(hop[j] < min)

min = hop[j];

}

hop[i] = min + 1;

}

}

}

return hop[0];

}

int main() {

int a[] = {1, 3, 6, 0, 0, 3, 2, 3, 6, 8, 9};

int n = minhops(a, 11);

if(n != INT\_MAX)

printf("%d\n", minhops(a, 11));

return 0;

}

Sol2 : Better

Since you can chose any x in [1,A[i]] I guess there is a pretty simple solution :

start at 0:

select the next reachable element from which you can reach the farther element. i.e chose i that maximize i+A[i+x] for x in [1,A[i]]

until you arrive at the end of the list.

Example:

{2 , 4 , 1 , 2 , 3 , 2 , 4 , 2}

start at 0

from 0 you can get to 1 or to 2:

from 1 you can get to 4

from 2 you can get to 3

therefore max(0+A[0+x]) is for i = 1

chose 1 from 1 you can get to 2 3 4:

from 4 you can get to 7

from 3 you can get to 5

from 2 you can get to 3

therefore max(1+A[1+x]) is for i = 4

chose 4

you can reach 7

stop

the resulting list is :

0,1,4,7

As explained in my comments I think it's O(N), because from i you reach i+x+1 in at least 2\*x operations.

## Given millions of time intervals, compute the maximum weights and the time when maximum weights occurred. Time interval is defined as followings (s: starting time, e: ending time, w: weight). And pls note that s, e, and w are not necessary integer.

suppose the interval is [1, 4] weight 2, [3, 7], weight 5, [8, 9], weight 3

sorted the boundaries of intervals to get array 1, 3, 4, 7, 8, 9

create weight array in this way: if the element of interval array is begin of some interval, put the positive of weight on weight array, if it is end of some interval, put negative weight on it. The we get 2, 5, -2, -5, 3, -3, then it converts to find maximum subarray problem. it is 2+5 = 7 in this example

Good solution. One special case is [0, 2][2, 4]. We definitely want to minus weight of ending point first before adding weight of starting point.

## There is a 3\*3 grid with 8 numbered tiles and 1 vacant space.Starting with initial grid arrangement, devise a strategy to reach final grid arrangement.

Operators: Move Blank square Left, Right, Up or Down.

Grid[3][3]={1,2,3, 0,4,5, 6,7,8};

Strategy:

One method is to represent all the states as graph nodes. And find a path from initial state to the final state.

Graph edges represents the movement of vacant space to one of the four possible moves.

# Math

Bitwise operations

Mathematical functions

Base conversion

## Output all dates "Fridays, 13th" in the format dd.mm.yyyy starting from 1st Jan 1900 (Monday)

void printAndCalculateDates(int& lastFriday, int& month, int& year, int numDays)

{

while (lastFriday <= numDays)

{

if (lastFriday == 13)

{

cout << "Date : " << lastFriday << "." << month << "." << year << endl;

}

lastFriday = lastFriday+7; // Increment the week

}

if (lastFriday > numDays)

{

lastFriday = lastFriday - numDays;

month = month+1;

//cout << "Date : " << lastFriday << "." << month << "." << year << endl;

if (month > 12)

{

month = 1;

year = year+1;

}

}

}

void printDate()

{

int currentDay = 10;

int currentYear = 2011;

int currentMonth = 11;

int lastFriday = 5;

int year = 1900;

int month = 01;

while ( (year <= currentYear) )

{

if ((year == currentYear) && (month == currentMonth))

{

if (lastFriday > currentDay)

{

break;

}

}

switch(month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

{

int numDays=31;

printAndCalculateDates(lastFriday, month, year, numDays);

break;

}

case 2:

{

int numDays=28;

if (year%4 == 0)

{

if(year%100==0)

{

if(year%400==0)

numDays = numDays+1;

}

else

numDays = numDays+1;

}

printAndCalculateDates(lastFriday, month, year, numDays);

break;

}

case 4:

case 6:

case 9:

case 11:

{

int numDays=30;

printAndCalculateDates(lastFriday, month, year, numDays);

break;

}

}

}

}

int main()

{

printDate();

return 0;

}

## There are three arrays of numbers A,B and C. You have to find out all tuples <a,b,c> such that a-b = c where a is from A,b is from B and c is from C.

that is, sort B and C, then for each element of A check if A[i] == B[j] + C[k]

using standard approach. Complexity: O(n^2)

void find\_sum(int \*a, int \*b, int \*c, int n) {

// check if a == b + c

for(int i = 0; i < n; i++) {

int x = a[i];

int l = 0, r = n - 1;

for(; ;) {

int s = b[l] + c[r];

if(s == x) {

printf("%d = %d + %d\n", x, b[l], c[r]);

l++;

} else if(s < x) {

l++;

} else

r--;

if(l == n || r == -1)

break;

}

}

printf("brute force check:\n");

for(int i = 0; i < n; i++) {

int x = a[i];

for(int j = 0; j < n; j++) {

for(int k = 0; k < n; k++) {

if(x == b[j] + c[k]) {

printf("all: %d = %d + %d\n", x, b[j], c[k]);

}

}

}

}

}

int main() {

int a[] = {8, 9, 10, 11, 20, 22};

int b[] = {1, 5, 6, 6, 9, 11};

int c[] = {3, 4, 7, 11, 12, 15};

int n = sizeof(a) / sizeof(int);

find\_sum(a, b, c, n);

return 1;

}

## Convert a double-precision number to rational, i.e.: 0.125 -> 1/8 don't care about arithmetic overflow

find the numerator remove the decimal we will get the numerator

here .125 numerator is 125

now find the denominator =10^(number of digit in numerator) here 10^3=1000

now find the HCF of numerator and denominator

here HCF of 125 and 1000 is 125

now divide the numerator and denominator by HCF

*#include <iostream>*

*template*< size\_t A, size\_t B >

*struct* GCD {

*enum* { value = GCD< B, A % B >::value };

};

*template*< size\_t A >

*struct* GCD<A,0> {

*enum* { value = A };

};

*int* main() {

std::cout << "GCD( 12, 18 ) = " << GCD<12, 18>::value << std::endl;

}

## A stream of 1's and 0's are comming .At any time we have to tell that the resultant number from the binary digits till that point is divisible by 3 or not .For eg: let's see one example.Let 1 come (not div by 3) .then 1 come so resultant binary number is 11(3) which is divisible by 3 , then 0 come make it to 110(3) which is divisible by 3, then 0 come make it to 1100(12) which also divisible by 3 .

Suppose we have to divide a number ie 1279 by 3. we get 1279 as streams ..first i get 1 ..1%3=1..1 is stored in rem.Now 2 comes, we do rem=(1\*10+2)%3=0.Now 7 is in teh stream,

rem=(rem\*10+7)%3=1, when 9 comes rem=(1\*10+9)%3=1. So we get at each step where the stream entered till that place is divisible by three or not.

Same is the case when stream cotains 0 and 1.Insteam of multiplying the rem by 10, multiply by 2.

//

// main.c

// DivisibleBy3

//

// Created by Srikant Aggarwal on 07/12/11.

// Copyright 2011 NSIT. All rights reserved.

//

#include <stdio.h>

int getBits()

{

int bit;

scanf("%d", &bit);

return bit;

}

int main (int argc, const char \* argv[])

{

int rem = 0;

int bit;

while(1)

{

bit = getBits();

rem = ((rem << 1) + bit)%3;

if(rem == 0)

printf("\n Divisible by 3 \n");

else

printf("\n Not divisible by 3 \n");

};

}

## Given n lines in the plane (for simplicity assume no 3 lines intersect at one point). Count the total number of triangles in the plane created by these lines. Observe that smaller triangles may be part of larger ones. Look here for example: <http://farm8.staticflickr.com/7021/6465828833_15e7447992_z.jpg>

## Design an algorithm to find the immediate greater number to a given number, such that the result has the same digits as the given input number.

Example:

Input Output

1234 1243

1243 1324

Take the last digit - keep comparing this with digits from the back (one by one) till you reach a digit that is lesser than the last digit. Now swap these two.

Now just sort the numbers in the portion that was compared (after the digit where the swap compared).

Store the number in a character array (num) and traverse back from the last. As soon as it is found that the digit at the current index (i) is less compared to the element to its right,we have to find the smallest number in the right part starting from i+1 which just exceeds num[i]. Swap that number with num[i] and sort the right array in ascending order.

e.g. For 1342

i = 1 since it is less than its next element to the right (which is 4)

We find the nearest digit in the right array starting from index 2

and in this case the number is 4.

Swap them

So now the number is 1432

Then we sort the right part to get 1423

Assume an array a[1,…,n].

1) Start from back, try to find an element a[k], where there is at least one element among a[k+1,…n] is larger than a[k].

2) Find the element a[h] among a[k+1,…n], which is the smallest element larger than a[k], then swap a[k] and a[h].

3) sort the new sub array a[k+1,…,n] in ascending order.

#include<cstdio>

#include<cstdlib>

#include<iostream>

#include<cstring>

using namespace std;

int searchJustBig(const char\* num, char c, int lpos, int rpos);

int comp(const void\* a, const void\* b)

{

char p = \*(char\*)a;

char q = \*(char\*)b;

if(p<q)

return -1;

else if(p>q)

return 1;

else

return 0;

}

int main()

{

cout<<"Enter the number ";

char num[1024];

scanf("%s", num);

unsigned len = strlen(num);

int i = len-2;

char tmp = num[len-1];

for(; i>=0; i--)

{

if(num[i] < tmp)

{

int val = searchJustBig(num, num[i], i+1, strlen(num)-1);

swap(num[val], num[i]);

qsort(num+i+1, strlen(num)-i-1, sizeof(char), comp);

break;

}

else

{

tmp = num[i];

}

}

if(i == -1)

printf("NONE\n");

else

printf("%s\n", num);

}

int searchJustBig(const char\* num, char c, int lpos, int rpos)

{

if(lpos<rpos)

{

int mid = (lpos + rpos)/2;

if(num[mid] <= c)

return searchJustBig(num, c, lpos, mid);

else

return searchJustBig(num, c, mid+1, rpos);

}

else

{

if(num[lpos] > c)

return lpos;

return lpos-1;

}

}

## Write a function that takes an integer and returns a char array that contains the -2 base representation of the given integer.

Example:

Input Output

7 11011

3 111

2 110

void negativeBaseRepresentation(int num)

{

char cNum[100]={0};

char\* pcNum=&cNum[98];

int remainder=0, quotient=0;

int sign = 1;

int base = 2;

do

{

remainder = num%base;

if(sign==1)

{

num = num/base;

quotient=num;

}

else

{

num = (num+1)/base;

}

sign=0-sign;

\*pcNum-- = remainder + '0';

}while(quotient != 0);

printf("Binary: %s\n", pcNum+1);

}

## Write a function to check if an integer is square or not? Ex, 49 is square, 48 is not.

#include

int sqrtof\_perfect(int start,int end,int n)

{

if(start>end)

return 0;

while(start<=end)

{

int mid=(start+end)/2;

//int temp=mid\*mid;

if(mid\*mid==n)

return mid;

else if(mid\*mid>n)

end=mid-1;

else

start=mid+1;

}

return -1;

}

## given an array , find three numbers a,b and c such that a^2+b^2=c^2 .. I have given algo of o(n^2) complexity. But he is expecting still better algo.

sort(S);

for i=0 to n-3 do

a = S[i];

k = i+1;

l = n-1;

while (k<l) do

b = S[k];

c = S[l];

if (a^2+b^2-c^2 == 0) then

output a, b, c;

exit;

else if (a^2+b^2 < c^2) then

l = l - 1;

else

k = k + 1;

end

end

end

## Given a number N find the next number M which is greater than N and the difference between M and N is small and M consists of same digits as N ! i.e M is an anagram of N ...

Example : 678 -- Ans=687

52430 -- Ans=53024

if number is 876 ans=no solution

if the numbers are all in descending order

return no solution.

Else walk from the back and find the first number that violates descending order.

in this case it would (24)... Look at the remaining numbers (30).

Replace 2 by the next number that is higher than 2 in ( 430)

So you get 53... sort the remaining numbers in ascending order ( 024)

You end up with 53024

## Given a number, convert it to minimal sum of squares. N can be represent as “x^2 + y^2 + z^2” or “a^2 + b^2”. But here the exception is minimal i.e. the answer will be “a^2 + b^2”

<http://www.careercup.com/question?id=10317685>

<http://stackoverflow.com/questions/3967769/represent-natural-number-as-sum-of-squares-using-dynamic-programming>

## Given a big unsorted list of 64-bit integers, find an element not in list

If the list contains all 64-bit numbers except only ONE number then XOR all the numbers will give the missing number.

## Itoa

/\* itoa: convert n to characters in s \*/

void itoa(int n, char s[])

{

int i, sign;

if ((sign = n) < 0) /\* record sign \*/

n = -n; /\* make n positive \*/

i = 0;

do { /\* generate digits in reverse order \*/

s[i++] = n % 10 + '0'; /\* get next digit \*/

} while ((n /= 10) > 0); /\* delete it \*/

if (sign < 0)

s[i++] = '-';

s[i] = '\0';

reverse(s);

}

## How do you convert a decimal number to its hexa-decimal equivalent.Give a C code to do the same

int main()

{

int n,r[10],i=0,number,j=0;

printf("Enter the decimal number\n");

scanf("%d",&number);

while(number>0)

{

r[i]=number%16;

number=number/16;

i++;

j++;

}

cout<<"The hexa decimal equivalent is ";

for(i=j-1;i>=0;i--)

{

if(r[i]==10)

printf("A");

else if(r[i]==11)

printf("B");

else if(r[i]==12)

printf("C");

else if(r[i]==13)

printf("D");

else if(r[i]==14)

printf("E");

else if(r[i]==15)

printf("F");

else

printf("%d",r[i]);

}

printf("\n");

system("pause");

}

## Print if a given integer is a palindrome or not

n = num;

rev = 0;

while (num > 0)

{

dig = num % 10;

rev = rev \* 10 + dig;

num = num / 10;

}

cout << "Number " << (n == rev ? "IS" : "IS NOT") << " a palindrome" << endl;

## How one would implement int Power?

int ipow(int base, int exp)  
{  
    int result = 1;  
    while (exp)  
    {  
        if (exp & 1)  
            result \*= base;  
        exp >>= 1;  
        base \*= base;  
    }  
  
    return result;  
}

## Given a rectangle of dimensions b\*h, and a input no N, find the size(side length) a such that N squares of size a can fit into the rectange. Optimize based on the minimum wasted of space in the rectangle b\*h

1st condition is l < max(b/N, h/N) since we can always put all squares in one line . it gives you max possible size of square for cases where either b > Nh or h > bN.

2nd condition is l < sqrt (b\*h/N)

3rd condition is l < min(b, h)

## Given a number, convert it to minimal sum of squares.

N can be represent as “x^2 + y^2 + z^2” or “a^2 + b^2”. But here the exception is minimal i.e. the answer will be “a^2 + b^2”.

<http://www.careercup.com/question?id=10317685>

<http://stackoverflow.com/questions/3967769/represent-natural-number-as-sum-of-squares-using-dynamic-programming>

## Given 3 prime numbers and an integer k, find the kth number if all the nos which are having these 3 prime numbers as their factors are arranged in increasing order.

Eg. prime numbers - 2,3,5

The increasing sequence will be 2,3,4,5,6,8,9...

we can use heap.  
The heap size can be at most 1500.  
Originally heap will have 1 only.  
Every step, we increment count, remove heap\_min and insert heap\_min\*2 and heap\_min\*3 and heap\_min\*5.  
Note that the heap\_min is the next ugly number.  
When the count reaches 1500 we stop.

<http://discuss.techinterview.org/default.asp?interview.11.541169.22>

<http://perl.plover.com/Stream/stream.html#streams>

<http://www.geeksforgeeks.org/archives/753>

# include<stdio.h>

# include<stdlib.h>

# define bool int

/\* Function to find minimum of 3 numbers \*/

unsigned min(unsigned , unsigned , unsigned );

/\* Function to get the nth ugly number\*/

unsigned getNthUglyNo(unsigned n)

{

unsigned \*ugly =

(unsigned \*)(malloc (sizeof(unsigned)\*n));

unsigned i2 = 0, i3 = 0, i5 = 0;

unsigned i;

unsigned next\_multiple\_of\_2 = 2;

unsigned next\_multiple\_of\_3 = 3;

unsigned next\_multiple\_of\_5 = 5;

unsigned next\_ugly\_no = 1;

\*(ugly+0) = 1;

for(i=1; i<n; i++)

{

next\_ugly\_no = min(next\_multiple\_of\_2,

next\_multiple\_of\_3,

next\_multiple\_of\_5);

\*(ugly+i) = next\_ugly\_no;

if(next\_ugly\_no == next\_multiple\_of\_2)

{

i2 = i2+1;

next\_multiple\_of\_2 = \*(ugly+i2)\*2;

}

if(next\_ugly\_no == next\_multiple\_of\_3)

{

i3 = i3+1;

next\_multiple\_of\_3 = \*(ugly+i3)\*3;

}

if(next\_ugly\_no == next\_multiple\_of\_5)

{

i5 = i5+1;

next\_multiple\_of\_5 = \*(ugly+i5)\*5;

}

} /\*End of for loop (i=1; i<n; i++) \*/

return next\_ugly\_no;

}

/\* Function to find minimum of 3 numbers \*/

unsigned min(unsigned a, unsigned b, unsigned c)

{

if(a <= b)

{

if(a <= c)

return a;

else

return c;

}

if(b <= c)

return b;

else

return c;

}

/\* Driver program to test above functions \*/

int main()

{

unsigned no = getNthUglyNo(150);

printf("%dth ugly no. is %d ", 150, no);

getchar();

return 0;

}

# Data Structures

Hashtables

Stacks

Smart pointers

Bitsets

## Give an algorithm for finding duplicate parenthesis in a expression. (( a + b ) \* (( c + d )))

1) take a stack S

2) Keep pushing chars from input in S, one by one

3) if charFromInput == ')' , bool check = true and while(S.pop() != '(');

4) if check == true && nextCharFromInput == ')' && S.pop() == '('

"Double parantheses detected."

5) Else check = false; continue;

## Code up a system that will accept a series of telephone keypresses and return a list of possible names from a supporting data structure. Describe both the data structure and the insert and search methods

## Write efficient code for Singleton class. I wrote the double-checked locking version.

## Write a C program to create a bitmap of any size as determined by user. Say user says 64k bitmap, and then create 64k long bitmap. Have set and unset methods.

## Simple Reference counted smart pointer

#ifndef smart\_pointer\_H  
#define smart\_pointer\_H  
  
template < typename T > class smart\_pointer  
{  
    private:  
        T\*    pointer;        
        int reference\_count;      
  
    public:  
  
        smart\_pointer() : pointer(0), reference\_count(-1) {}  
  
        smart\_pointer(T\* p) : pointer(p)  
        {  
            if (p != NULL)  
            {  
                this->reference\_count = 1;  
            }  
  
            else  
            {  
                this->reference\_count = -1;  
            }  
        }  
  
        smart\_pointer(const smart\_pointer <T> & p) : pointer(p.pointer),     reference\_count(p.reference\_count + 1) {}  
        bool operator == (const smart\_pointer <T>& p) { return pointer == p.pointer; }  
        bool operator != (const smart\_pointer <T>& p) { return pointer != p.pointer; }  
  
  
        ~ smart\_pointer()  
        {  
            if(-- reference\_count == 0)  
        {  
                std::cout << "Destructing: " << '\n';  
                delete pointer;  
            }  
        }  
  
        T& operator \*  () { return \*pointer; }  
        T\* operator -> () { return pointer; }  
  
        smart\_pointer <T> & operator = (const smart\_pointer <T> & p)  
        {  
                if (this != &p)  
                {  
                    if( -- reference\_count == 0)  
                    {  
                        delete pointer;  
                    }  
  
                        pointer = p.pointer;  
                        reference\_count = p.reference\_count + 1;  
                }  
  
        return \*this;  
        }  
};

## Find the element with the middle value among all elements of the stack...you can you extra space but complexity should be minimum. Problem is to find the median of all the elements in the stack.

* Pop all the elements in an array and sort the array, get the median and return it.

I need a solution that is not O(n3). I gave the O(n3) solution but the interviewer was not happy. We also cannot copy the input vector as space requirements in O(1).

A zero-indexed array A consisting of N integers is given. A triplet (P, Q, R) is triangular if and

A[P] + A[Q] > A[R],  
A[Q] + A[R] > A[P],  
A[R] + A[P] > A[Q].

For example, consider array A such that

A[0] = 10 A[1] = 2 A[2] = 5  
A[3] = 1 A[4] = 8 A[5] = 20  
Triplet (0, 2, 4) is triangular.

public int triangle(int[] A)

that, given a zero-indexed array A consisting of N integers, returns 1 if there exists a triangular triplet for this array and returns 0 otherwise.

Assume that:

N is an integer within the range [0..100,000];  
each element of array A is an integer within the range[-2,147,483,648..2,147,483,647].  
For example, given array A such that

A[0] = 10 A[1] = 2 A[2] = 5  
A[3] = 1 A[4] = 8 A[5] = 20  
the function should return 1, as explained above. Given arrayA such that

A[0] = 10 A[1] = 50 A[2] = 5  
A[3] = 1  
the function should return 0.  
Expected worst-case time complexity: O(n log n)  
Expected worst-case space complexity: O(1)

a) sort it descending in place  
b) iterate looking for a[i], a[i + 1], a[i + 2] (p, q, r) such that p+q > r etc...

public class FindTriplets {

public static void findTriplets(int[] input){  
 if (input.length < 3){  
 System.out.println("Array too small");  
 return;  
 }  
   
 //Tuned quicksort n\*log(n)  
 Arrays.sort(input);  
   
 for (int i=0; i+2<input.length;i++){  
 if (input[i] + input[i+1] > input[i+2]){  
 System.out.println("Found a triple:");  
 System.out.println("A " + input[i]);  
 System.out.println("B " + input[i+1]);  
 System.out.println("C " + input[i+2]);   
 }   
 }  
 }  
   
 public static void main(String[] args){  
 int[] input = new int[]{10,2,5,1,8,50};  
 findTriplets(input);  
 }  
   
}

## Code/Define Algorithm to find if a given string has balanced parentheses, where we have 3 types 1. ( ) 2. [ ] 3. { }.

\*the soln is to use a stack   
\*and push every opening bracket into the stack,   
\*and every closing bracket should pop from the stack  
\*the closing bracket should make a pair with the popped value from stack  
\*at no time the stack should go negative.  
\*at end the stack should be empty.

## Write code to retrieve max-valued element from a stack. Time complexity must be O(1).

One way to do so is by keeping another stack, containing max elements and updating it for every push and pop operation.

## Given an infix expression convert into postfix

make use of shuntington algorithm.

you see a operand you pass it to output.

you see a operator:

check the priority of operator on top of stack:

if higher priority push it to the stack

else (lower or equal priority) pop the operators until the one with lower priority is found

repeat till the end of the infix expression.

String toPostfix(String infixStr){

if( infixStr == null ){

throw new IllegalArgumentException("NULL str passed");

}

final StringBuilder postfixBuf = new StringBuilder( infixStr.length() );

final Deque<Character> operatorsStack = new LinkedList<Character>();

for( char ch : infixStr.toCharArray() ){

// operator found

if( isOperator(ch) ){

while( ! operatorsStack.isEmpty() ){

char prevOperator = operatorsStack.pop();

if( compareOperators(prevOperator, ch ) < 0 ){

operatorsStack.push(prevOperator);

break;

}

postfixBuf.append( prevOperator );

}

operatorsStack.push(ch);

}

// operand found

else{

postfixBuf.append( ch );

}

}

while( ! operatorsStack.isEmpty() ){

postfixBuf.append( operatorsStack.pop() );

}

return postfixBuf.toString();

}

## Write code to retrieve max-valued element from a stack. Time complexity must be O(1).

**Two stack solution:**

The stack solution will work because every time you push number into first stack, we will push into second stack only if this stack's top element is less than number being pushed into first stack and while popping we will do it in second stack only if the second stack's top element is same as the element popped.

This way second stack maintains the current max till the current max was popped and exposes the next max in the stack.

# Design

## I have four operations, Set(i) sets the ith bit, Unset(j) unsets the jth bit, Set(i,j) sets the bits b/w i and j, Unset(i,j) unsets the bits b/w i and j. Design a data structure for this problem with minimum time and space complexity.

## Design the system for threater reservation system. for example Seat or chairs are organized in the form of rows and columns. When the first person come and book the ticket need to provide a seat on the middle of the last row. When next person come we have to provide empty space between the existing audience and book the ticket for the set of people. How we will design this system?

## Given the classes below how would you improve the class design and their relationships

Furniture  
SteelChair  
SteelTable  
WoodenChair  
WoodenTable

Furniture is the interface.

Chair & Table extends Furniture

WC and SC extends Chair

WT and WT extends Table

## Given a Parking lot having fixed number of slots, where a car can enter the lot if there is a free slot and then it will be given the direction of the free slot. When exits the car has to pay the fees for the duration of the time the car is in the slot. How you will design the data model and functional model.

ParkingLotCollection

ParkingLot

SlotID

IsFree()

GetDirection()

UpdateSlot(free/occupied,CarID)->update the slot status and start the timer.

Car

CarID

ParkingLot \*lot;

TimeOccupied()

## Design classes to calculate the price of a pizza with different crusts, toppings and sauces

## Design and create a Collections library. Make it flexible from two aspects

## Assume you have a monolithic web site and you are asked to rearchitect the website

## OO Design for a restaurant reservation system.

## Design a DFS that supports mount, open, read, write, close and unmount

## How would you architect an online website?

## Suppose you are writing a crawler meant to crawl the entire web. How would you avoid reindexing the same page.

## Given a database table of order entries and n service instances (S0..Sn-1) processing these entries. Instance Si processes a record if the order-id%n = i. Do you see any issues with this design?

## Given a series of stock prices in an array. Pick when to buy and when to sell to gain max profit.

## Discussion on various data structure where the following operations could have minimal cost:

1.insert

2.delete

3.search

4.return any element

Hash

## The best data structure to store the phone book. Or dictionary

<http://en.wikipedia.org/wiki/Trie>

## Design an API for this scenario. There is a onetime input of a set of data. The user works on an interface that has two options- a) Find(x) which returns x if found. b) Findkthmin() This interface is used several hundred times a day. Describe the data structure& algorithm that you will use.

The API will have two calls: find(x) and kthmin();

For find(x), we can keep a hash table which hashes the one time input to proper keys, and then everytime find(x) is called it takes expected O(1) lookup time for answering.

For kthmin(x), we sort the array one time, and then answer kth min O(1) time.

## A "Most efficient data structure" is designed to optimize the following operations. Pop, Push, Min. The best possible time-complexities with no extra space, respectively would be?

May be min. heap.

Push & Pop - O(log n)

Min - O(1).

## You need to implement two functions of hashing int get(int key) // will return value at key void put (int key,int value) // put the vaue at index key, both functions should be of order O(1) and there is an upper limit n on size, such that at the max only n elements will get stored and if there is a conflict then you need replace the least recently used element

Hashtable + Linked list. Typical implementation of LRU caches.

## Design an algorithm to perform operation on an array Add(i,y)-> add value y to i position sum(i) -> sum of first i numbers we can use additional array O(n) and worst case performance should be O(log n) for both operation

<http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=binaryIndexedTrees>

<http://comeoncodeon.wordpress.com/2009/09/17/binary-indexed-tree-bit/>

<http://www.algorithmist.com/index.php/Fenwick_tree>

#include <vector>

using namespace std;

// In this implementation, the tree is represented by a vector<int>.

// Elements are numbered by 0, 1, ..., n-1.

// tree[i] is sum of elements with indexes i&(i+1)..i, inclusive.

// (Note: this is a bit different from what is proposed in Fenwick's article.

// To see why it makes sense, think about the trailing 1's in binary

// representation of indexes.)

// Creates a zero-initialized Fenwick tree for n elements.

vector<int> create(int n) { return vector<int>(n, 0); }

// Returns sum of elements with indexes a..b, inclusive

int query(const vector<int> &tree, int a, int b) {

if (a == 0) {

int sum = 0;

for (; b >= 0; b = (b & (b + 1)) - 1)

sum += tree[b];

return sum;

} else {

return query(tree, 0, b) - query(tree, 0, a-1);

}

}

// Increases value of k-th element by inc.

void increase(vector<int> &tree, int k, int inc) {

for (; k < (int)tree.size(); k |= k + 1)

tree[k] += inc;

}

## Design a data structure for a phone address book with 3 fields name, phone number , address.

One must be able to search this phone book on any of the 3 fields.

Maintain three hashes for this.

Map<string,string>AddressMap;

Map<string,AddressMap::iterator> NameMap;

Map<string, NameMap::iterator> PhoneMap;

Or we can create three suffix trees for the searching and store records in some database.

## Design pizza topping

Ch03.pdf

Decorator pattern

# Puzzles

## There are 100 prisoners, and a officer of them. Now the officer gave the command to the prisoner that next day they will be going to wear a hat which they will not be know its color. But its color will be either Red or Blue. And he says that the entire prisoner will be standing in a line. And then the officer will start asking the color of the prisoner one by one from the back. Whichever prisoner says the wrong color of his hat, gets shoot .So now we have to find out what strategy should the prisoners should apply to safe maximum prisoners.

Last one speaks the odd number caps he sees (7R, 92B, 1his)—R

Now say he survives (8R, 92B)

Next one sees either (7R, 91B) or (6R, 92B)🡪 in first he is wearing B in other R.

## Given a set of cubes with sides colored with random colors. Find if the cubes can be stacked one on top of other such that each of the four lateral sides has the same vertical color.

For a cube there are six faces , assume we have used 3 colors entirely(GREEN,RED,BLUE) .

For ex : any cube will have the following edges

TOP DOWN SIDE1 SIDE2 SIDE3 SIDE4

Cube1 GREEN RED BLUE BLUE RED GREEN

Cube2 RED RED BLUE BLUE RED BLUE

to stack around we need to select a color for the vertical wall .

so the bottom cube four sides decides the how much height we can build the cubes stack .

Now we need arrange the colors of the cubes, and find out the 4 color combination which is contained in maximum number of

cubes .

Take a Hash Map and store each color combination in the Hash Map and increment its count for every hit. The color with maximum

number of hits is the color combination which we need to choose.

One point we need to note here is for a given cube with colors there are three possible orderings , visualize it you will know

Possible Orderings are

TOP DOWN SIDE1 SIDE2 SIDE3 SIDE4

Cube1 GREEN RED BLUE BLUE RED GREEN

BLUE RED GREEN BLUE RED GREEN - Exchanged side1 and side 3 as top and bottom

BLUE GREEN BLUE GREEN RED RED - Exchanged side2 and side 4 as top and bottom

So if cube 1 is taken as bottom most cube SIDE1 SIDE2 SIDE3 SIDE4 colors become the vertical walls and we need to check

for cubes which has same color combination as cube 1.(we can reorder the colors among the walls)

Hash Map

1)key SIDE1 SIDE2 SIDE3 SIDE4

BLUE BLUE RED GREEN

value 10

2)key SIDE1 SIDE2 SIDE3 SIDE4

RED BLUE RED GREEN

value 5

Once we populate everything to HashMap we can find the color combination which is repeated maximum no of times and stack the cubes

## Given n red balls and m blue balls and some containers, how would you distribute those balls among the containers such that the probability of picking a red ball is maximized, assuming that the user randomly chooses a container and then randomly picks a ball from that.

Put 1 red balls in all the containers except one in which we will put all the balls.

## Given a cube of size n\*n\*n (i.e made up of n^3 smaller cubes), find the number of smaller cubes on the surface. Extend this to k-dimension.

For 3 it is 26.

K^3 - (k-2)^3

## 100 rooms in a prison. All doors are closed initially. A policeman on round i will toggle the doors that are in multiples of i. After 100 rounds, what doors will be open?

Initially all doors are closed  
After 100 iterations a door will be open if it has been flipped odd number of times else it will be closed   
A door will get flipped in ith iteration if i is a factor of the door number n  
which means door number which have odd number of factors will be left open after 100 iterations

Mouse Hover Me =>Perfect squares will have odd number of factors. so open doors are 4, 9, 16..

## Given a clock, take time as input in any format; provide a formula to calculate the angle between the minute and hour hand.

HH:MM 0..11:0..59

HHAng=HH\*(360/12)+MM\*(360/(60\*12))

MMAng = MM\*(360/60)

AngDiff = Abs(HHAng-MMAng)

## There are 25 horses among which you need to find out the fastest 3 horses. You can conduct race among at most 5 to find out their relative speed. At no point you can find out the actual speed of the horse in a race. Find out how many races are required to get the top 3 horses.

It requires 8 Iterations in total: (divide horses in to 5 groups: A,B,C,D,E)  
Race(1 - 5): Race 5 horses each group and let Winner horses are A1,B1,C1,D1,E1.  
Race-6: Race A1,B1,C1,D1,E1. This will give the fastest horse. Let say C1 is the fastest horse.  
Race-7: The second fastest could be the second fastest in C group (C2) or fastest of other groups. So race A1,B1,C2,D1,E1. Let say B1 won. So B1 is the second fastest of all.  
Race-8: The third fastest could be the second fastest of in B group (B2) as B1 is the over all second fastest horse. So race A1,B2,C2,D1,E1.

## There are n petrol bunks arranged in circle. Each bunk is separated from the rest by a certain distance. You choose some mode of travel which needs 1litre of petrol to cover 1km distance. You can't infinitely draw any amount of petrol from each bunk as each bunk has some limited petrol only. But you know that the sum of litres of petrol in all the bunks is equal to the distance to be covered. ie let P1, P2, ... Pn be n bunks arranged circularly. d1 is distance between p1 and p2, d2 is distance between p2 and p3. dn is distance between pn and p1.Now find out the bunk from where the travel can be started such that your mode of travel never runs out of fuel.

This is being made more complicated than it is.  (I didn't even try to parse it.)  Here's a simple O(n) solution.  Start from an arbitrary node (which we shall call node 0) and decide on an arbitrary direction (clockwise or counterclockwise).  Set fuel=0 and repeat: add the amount in the current node; subtract the distance to the next node; proceed to the next node.  Stop upon return to node 0.  Let i be the node where upon arrival fuel was minimum (possibly negative).  The answer is node i.  
Proof.  Let x denote the value of fuel upon entry to node i.  It is easy to see that if we start the above process from node i (in the same direction), the value of fuel at any given node will be the value we calculated originally minus x.  (Note that this is true also for the nodes in the range 0..i because going full circle gives 0.)  Since x is minimum among the values calculated originally, all newly calculated values will be non-negative, i.e., the solution is feasible.

# Miscellaneous

## Write a function to reverse a UTF-8 encoded string in-place.

## There is a document and some key words are given. Find the minimum length of substring in the document which contains all the keywords. The substring can contain key word in any order.

## There is a straight road with 'n' number of milestones. You are given an array with the distance between all the pairs of milestones in some random order. Find the position of milestones.

Example:

Consider a road with 4 milestones(a,b,c,d) :

a <--- 3Km --->b<--- 5Km --->c<--- 2Km --->d

Distance between a and b = 3

Distance between a and c = 8

Distance between a and d = 10

Distance between b and c = 5

Distance between b and d = 7

Distance between c and d = 2

All the above values are given in a random order say 7, 10, 5, 2, 8, 3.

The output must be 3,5,2 or 2,5,3

## Given a large list of 2D Points (x,y) find the k closest points to a target P(x,y).

- Implement the solution

- Explain the running time

- Explain how you can optimize your code

- Explain some decisions behind why you chose to write your code that way.

Use formula- sqrtroot((x2-x1)^2)+(y2-y1)^2)) to find distance

find distance between every point in a list from given point.

Maintain max heap of k elements and replace max element with smaller incoming values apply heapify downward so that at any point of time you have k minimum values.

complexity:nLogK

## Given 0\*1\*, find index of first 1. Ie Given an infinite size array with only 0s and 1s and sorted. find the transition point where 0s end or 1s start

Binary Search.

A is the input array

1. start=0, end=1, mid=0

2. if((A[start]==0)&&(A[end]==1)) return start;

3. while(A[end] != 1)

start=end;

end=end<<1;

4. while(A[start+1] != 1)

mid=(start+end)/2;

if(A[mid]==0)

start=mid;

else

end=mid;

5. return start;

## Write a function which takes as parameters one regular expression(only ? and \* are the special characters) and a string and returns whether the string matched the regular expression.

## Given int n, the total number of players and their skill-point. Distribute the players on 2 evenly balanced teams.

Pick up the highest skilled player and put him in first team. Then next highest skilled player put him in second team.

After this take a difference of strength. Pick up next player and put him in the weak team.

Two variables, teamAStrength, teamBstrength,

Diff=teamAstrength-teamBstrength;

If(diff<0)

teamAstrength+= nextplayer;

else

teamBstrength+=nextplayer;

# Theory

## Under which conditions the default assignment operator is not generated by the compiler for your class ?

(assuming that you do not declare the assignment operator yourself)

A member class has const members.  
A member class has reference members.  
A member class or its base class has a private assignment operator (operator=).  
A base class or member class has no assignment operator (operator=).

## What is the time and space complexities of merge sort and when is it preferred over quick sort?

O(n) space for merging

O(n logn) time

Prefer merge sort over quick sort: when the elements are almost sorted. Quick sort worst time is O(n2)

when the data set is huge and is stored on external devices such as a hard drive, merge sort is the clear winner in terms of speed.

Stable sort.

## What is Map and what are the different ways of implementing map. Give pros and cons of each approach

Array

Hashing

Trees Binary trees

## Difference btw process and thread?

Process and threads are both units of execution that parallelize resource utilization.

Processes are independent execution units that contain their own state information, use their own address spaces, and only interact with each other via interprocess communication mechanisms

A thread is a more of a coding construct, a smaller unit of execution, a thread has its own

 Thread ID

 set of registers, stack pointer

 stack for local variables, return addresses

 signal mask

 priority

Threads share

* Process instructions
* Most data
* open files (descriptors)
* signals and signal handlers
* current working directory
* User and group id

## What is a tree data structure

A tree is a is finite non-empty set of elements in which one element is called the root and the remaining elements are partitioned into m>=0 disjoint subsets, each of which is itself a tree.

A binary tree is a finite set of elements that is either empty or is portioned into three disjoint subsets. The first subset contains single element called the root of the tree. The other two subsets are themselves binary trees.

## difference between tree and graph

A graph consists of a set of nodes and a set of edges. Each arc in a graph is specified by a pair of nodes.

Differences:

Trees are acyclic. Graphs can be cyclic

A tree can be described as a specialized case of graph with no self loops and circuits.

## We have a text file, how to choose between a hash-based index and a tree-based index?

Hash based needs good hashing function. Tree based doesn’t.

Disk usage in less in tree based index.

Tree based index will need more space. Hash based doesn’t need more space.

## Fork vfork exec

### Fork

fork() creates a child process that differs from the parent process only in its PID and PPID, and in the fact that resource utilizations are set to 0. File locks and pending signals are not inherited.

Under Linux, fork() is implemented using **copy-on-write pages**, so the only penalty that it incurs is the time and memory required to duplicate the parent’s page tables, and to create a unique task structure for the child.

**Returns**

On success, the PID of the child process is returned in the parent’s thread of execution, and a 0 is returned in the child’s thread of execution. On failure, a -1 will be returned in the parent’s context, no child process will be created, and errno will be set appropriately.

EAGAIN fork() cannot allocate sufficient memory to copy the parent’s page tables and allocate a task structure for the child.

EAGAIN It was not possible to create a new process because the caller’s RLIMIT\_NPROC resource limit was encountered. To exceed this limit, the process must have either the CAP\_SYS\_ADMIN or the CAP\_SYS\_RESOURCE capability.

ENOMEM fork() failed to allocate the necessary kernel structures because memory is tight.

### Vfork

vfork() differs from fork() in that the parent is **suspended** until the child makes a call to execve(2) or \_exit(2). The child shares all memory with its parent, including the stack, **until** execve() is issued by the child. The child must not return from the current function or call exit(), but may call \_exit().

Under Linux, fork() is implemented using copy-on-write pages, so the only penalty incurred by fork() is the time and memory required to duplicate the parent’s page tables, and to create a unique task structure for the child. **However, in the bad old days a fork() would require making a complete copy of the caller’s data space, often needlessly, since usually immediately afterwards an exec() is done. Thus, for greater efficiency, BSD introduced the vfork() system call, that did not fully copy the address space of the parent process, but borrowed the parent’s memory and thread of control until a call to execve() or an exit occurred.** The parent process was suspended while the child was using its resources. The use of vfork() was tricky: for example, not modifying data in the parent process depended on knowing which variables are held in a register.

### Exec

The functions are declared in the unistd.h header for the POSIX standard and in process.h for DOS, OS/2, and Windows.

int execl(char const \*path, char const \*arg0, ...);

int execle(char const \*path, char const \*arg0, ..., char const \* const \*envp);

int execlp(char const \*file, char const \*arg0, ...);

int execv(char const \*path, char const \* const \* argv);

int execve(char const \*path, char const \* const \*argv, char const \* const \*envp);

int execvp(char const \*file, char const \* const \*argv);

Some implementations provide these functions named with a leading underscore (e.g. \_execl).

Function names

The base of each is exec (execute), followed by one or more letters:

e - An array of pointers to environment variables is explicitly passed to the new process image.

l - Command-line arguments are passed individually to the function.

p - Uses the PATH environment variable to find the file named in the path argument to be executed.

v - Command-line arguments are passed to the function as an array of pointers.

-1 on failure, with errno set to:

E2BIG The argument list exceeds the system limit.

EACCES The specified file has a locking or sharing violation.

ENOENT The file or path name not found.

ENOMEM Not enough memory is available to execute the new process image.

## IPC Linux

### Pipes

### Shared Memory Linux

In summary

A process creates a shared memory segment using shmget()|. The original owner of a shared memory segment can assign ownership to another user with shmctl(). It can also revoke this assignment. Other processes with proper permission can perform various control functions on the shared memory segment using shmctl(). Once created, a shared segment can be attached to a process address space using shmat(). It can be detached using shmdt() (see shmop()). The attaching process must have the appropriate permissions for shmat(). Once attached, the process can read or write to the segment, as allowed by the permission requested in the attach operation. A shared segment can be attached multiple times by the same process. A shared memory segment is described by a control structure with a unique ID that points to an area of physical memory. The identifier of the segment is called the shmid. The structure definition for the shared memory segment control structures and prototypews can be found in <sys/shm.h>.

int shmget(key\_t key, size\_t size, int shmflg);

int shmctl(int shmid, int cmd, struct shmid\_ds \*buf);

**SHM\_LOCK**

-- Lock the specified shared memory segment in memory. The process must have the effective ID of superuser to perform this command.

**SHM\_UNLOCK**

-- Unlock the shared memory segment. The process must have the effective ID of superuser to perform this command.

**IPC\_STAT**

-- Return the status information contained in the control structure and place it in the buffer pointed to by buf. The process must have read permission on the segment to perform this command.

**IPC\_SET**

-- Set the effective user and group identification and access permissions. The process must have an effective ID of owner, creator or superuser to perform this command.

**IPC\_RMID**

-- Remove the shared memory segment.

void \*shmat(int shmid, const void \*shmaddr, int shmflg);

int shmdt(const void \*shmaddr);

### Memory Map

* First open() the file, then
* mmap() it with appropriate access and sharing options

#include <sys/types.h>

#include <sys/mman.h>

caddr\_t mmap(caddr\_t addr, size\_t len, int prot, int flags, int fildes, off\_t off);

int munmap(caddr\_t \*addr, size\_t length);

The starting address for the new mapping is specified in **addr**. If addr is **NULL**, then the kernel chooses the address at which to create the mapping

The **prot** argument describes the desired memory protection of the mapping (and must not conflict with the open mode of the file). It is either PROT\_NONE or the bitwise OR of one or more of the following flags:

PROT\_EXEC Pages may be executed.

PROT\_READ Pages may be read.

PROT\_WRITE Pages may be written.

PROT\_NONE Pages may not be accessed.

The **flags** argument determines whether updates to the mapping are visible to other processes mapping the same region, and whether updates are carried through to the underlying file.

MAP\_SHARED Share this mapping. Updates to the mapping are visible to other processes that map this file, and are carried through to the underlying file. The file may not actually be updated until msync(2) or munmap() is called.

MAP\_PRIVATE Create a private copy-on-write mapping. Updates to the mapping are not visible to other processes mapping the same file, and are not carried through to the underlying file.

In addition, zero or more of the following values can be ORed in flags:

MAP\_ANONYMOUS The mapping is not backed by any file; its contents are initialized to zero.

MAP\_FIXED Don't interpret addr as a hint: place the mapping at exactly that address.

MAP\_GROWSDOWN Used for stacks. Indicates to the kernel virtual memory system that the mapping should extend downward in memory.

MAP\_LOCKED (since Linux 2.5.37) Lock the pages of the mapped region into memory

MAP\_NONBLOCK (since Linux 2.5.46) Only meaningful in conjunction with MAP\_POPULATE. Don't perform read-ahead: create page tables entries only for pages that are already present in RAM.

MAP\_NORESERVE Do not reserve swap space for this mapping. When swap space is reserved, one has the guarantee that it is possible to modify the mapping.

MAP\_POPULATE (since Linux 2.5.46) Populate (prefault) page tables for a mapping. For a file mapping, this causes read-ahead on the file. Later accesses to the mapping will not be blocked by page faults.

MAP\_UNINITIALIZED (since Linux 2.6.33) Don't clear anonymous pages. This flag is intended to improve performance on embedded devices.

The **munmap**() system call deletes the mappings for the specified address range, and causes further references to addresses within the range to generate invalid memory references.

int mlock(caddr\_t addr, size\_t len)

causes the pages in the specified address range to be locked in physical memory.

int munlock(caddr\_t addr, size\_t len)

releases the locks on physical pages. If multiple mlock() calls are made on an address range of a single mapping, a single munlock call is release the locks.

int mlockall(int flags) and int munlockall(void) are similar to mlock() and munlock(), but they operate on entire address spaces. mlockall() sets locks on all pages in the address space and munlockall() removes all locks on all pages in the address space, whether established by mlock or mlockall.

int msync(caddr\_t addr, size\_t len, int flags) causes all modified pages in the specified address range to be flushed to the objects mapped by those addresses. It is similar to fsync() for files.

long sysconf(int name) returns the system dependent size of a memory page. For portability, applications should not embed any constants specifying the size of a page. Note that it is not unusual for page sizes to vary even among implementations of the same instruction set.

int mprotect(caddr\_t addr, size\_t len, int prot) assigns the specified protection to all pages in the specified address range. The protection cannot exceed the permissions allowed on the underlying object.

int brk(void \*endds) and void \*sbrk(int incr) are called to add storage to the data segment of a process. A process can manipulate this area by calling brk() and sbrk(). brk() sets the system idea of the lowest data segment location not used by the caller to addr (rounded up to the next multiple of the system page size). sbrk() adds incr bytes to the caller data space and returns a pointer to the start of the new data area.

## Pthread

# Permutation and Combinations

## give a \*recursive\* algorithm to print all permutations of 'n' consecutive integers in lexicographic order, i.e.:

1 2 3 4

1 2 4 3

1 3 2 4

...

4 3 2 1

void permutate( char[] str, int index )  
{  
 int i = 0;  
 if( index == strlen(str) )  
 { // We have a permutation so print it  
 printf(str);  
 return;  
 }  
 for( i = index; i < strlen(str); i++ )  
 {  
 swap( str[index], str[i] ); // It doesn't matter how you swap.  
 permutate( str, index + 1 );  
 swap( str[index], str[i] );  
 }  
}

## Given an integer array. e.g. 1,2,3,4,5. Compute array containing elements 120,60,40,30,24 (2\*3\*4\*5,1\*3\*4\*5, 1\*2\*4\*5, 1\*2\*3\*5, 1\*2\*3\*4)

So I gave simple solution using division operator. He said that what if there is a 0 as an element and

what if there are two zeroes.

Then he asked me to do it without using division operator. So after little fumbling I did it in O(n)

extra space and O(n) time.

Then he asked me to do it in O(1) space and O(n) time.

## Find all the permutation of the given string? But take care of duplicate characters.

## given a set of letters and a length N, produce all possible output.(Not permutation). For example, give the letter (p,o) and length of 3, produce the following output(in any order you want, not just my example order)

ppp ppo poo pop opp opo oop ooo

another example would be given (a,b) and length 2

answer: ab aa bb ba

#include<stdio.h>

void print(char \*str, char \*pattern, int index)

{

if(index == 3)

{

printf("%s\n",str);

//\*(str+index) = '\0';

}

else

{ int i=0;

while(\*(pattern+i))

{

\*(str+index) = \*(pattern+i);

print(str,pattern,index+1);

i++;

}

}

}

int main()

{

char \*str = (char\*)malloc(sizeof(char)\*4);

\*(str+3)='\0';

char \*pattern = "012";

print(str,pattern,0);

getchar();

return 0;

}

## given a number, come up with all of the possible ways to insert '+' and '-' in that number. for example given 123, possible answer would be

1+23

1+2+3

1-23

1-2+3

1-2-3

1+2-3

12+3

12-3

void print(String head, String tail)

{

if (tail.length() == 0)

{

return;

}

char c = tail.charAt(0);

if (head.length() > 0)

{

System.out.println(head + "+" + tail);

print(head + "+" + c, tail.substring(1));

System.out.println(head + "-" + tail);

print(head + "-" + c, tail.substring(1));

}

print(head + c, tail.substring(1));

}

public void print(String numb)

{

print("", numb);

}

## Given a string "abcd" print combinations of length n. Example if n= 3 print abc, abd, acd, bcd. Was asked to use recursion.

# Yahoo 1-8 questions

## Given an 4n X 4n Matrix, where n is a positive integer taken as input. Imagine the matrix consisting of two interleaved coils whose centers are at the centre of the matrix. Implement a java program which takes an integer (n) as input and prints the two coils in two seperate lines.

Please have a look at the below examples to get a sense of what the two coils are :

• Example 1:

• Input: 1

• Matrix:

01 02 03 04

05 06 07 08

09 10 11 12

13 14 15 16

• Output the Two Coils as:

- Coil1: 10 06 02 03 04 08 12 16

- Coil2: 07 11 15 14 13 09 05 01

• Example 2:

• Input: 2

• Matrix:

01 02 03 04 05 06 07 08

09 10 11 12 13 14 15 16

17 18 19 20 21 22 23 24

25 26 27 28 29 30 31 32

33 34 35 36 37 38 39 40

41 42 43 44 45 46 47 48

49 50 51 52 53 54 55 56

57 58 59 60 61 62 63 64

• Output the Two Coils as:

- Coil1: 36 28 20 21 22 30 38 46 54 63 52 51 50 42 34 26 18 10 02 03 04 05 06 07 08 16 24 32 40 48 56 64

- Coil2: 29 37 45 44 43 35 27 19 11 12 13 14 15 23 31 39 47 55 63 62 61 60 59 58 57 49 41 33 25 17 09 01

package javaoutspacecode;

import java.util.Scanner;

/\*\*

\*

\* @author outspacecode AT gmail dot c0m

\*/

public class Coil

{

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

// TODO Auto-generated method stub

int n;

Scanner sca=new Scanner(System.in);

System.out.println("enter the value of n");

n=sca.nextInt();

int i,j,k;

int b[][]=new int[4\*n][4\*n];

int a=1;

for(i=0;i<;4\*n;i++)

{

for(j=0;j<;4\*n;j++)

{

b[i][j]=a;

a++;

System.out.print(b[i][j]);

System.out.print("\t");

}

System.out.println("");

}

int m = 8\*n\*n;

int c1[]=new int[m];

int c2[]=new int[m];

int p=0, p1=0, q1=2, nflg=1;

c1[p]= b[2\*n][2\*n-1];

int tmp = c1[p];

p++;

while( p < m )

{

for( p1 = 0; p1<q1; p1++)

{

c1[p]= tmp - 4\*n\*nflg;

tmp = c1[p];

p++;

if( p >= m )

{

break;

}

}

if( p >= m )

{

break;

}

for( p1 = 0; p1<q1; p1++)

{

c1[p]= tmp + nflg;

tmp = c1[p];

p++;

if( p >= m )

{

break;

}

}

nflg = nflg\*(-1);

q1+=2;

}

/\* get coil2 from coil1 \*/

for(i=0;i<;8\*n\*n;i++)

{

c2[i] = 16\*n\*n + 1 -c1[i];

}

System.out.println();

System.out.print("Coil1:");

for(i=0;i<;8\*n\*n;i++)

{

System.out.print(c1[i]);

System.out.print("\t");

}

System.out.println();

System.out.print("Coil2:");

for(i=0;i<;8\*n\*n;i++)

{

System.out.print(c2[i]);

System.out.print("\t");

}

}

}

## Suppose there is an array with numbers : 1, 14, 5, k, 4, 2, 54, k, 87, 98, 3, 1, 32. Output for this can be assuming k =20 1,14,5,4,2,3,1,k,k,54,87,98,32. Now sort this array in a way all k are in middle and all values on left of k are smaller (in any order) and on right are larger (in any order) Note: k is an integer value within range of 1 – 32768

See internal question: [Dutch National flag problem](#_Dutch_National_flag)

public void threeWayPartition(int[] a, int pivot)

{

int p = 0;

int q = a.length - 1;

for(int i = 0; i <= q;) {

if(a[i] < pivot) swap(a, i++, p++);

else if(a[i] > pivot) swap(a, i, q--);

else i++;

}

}

private void swap(int[] a, int i, int j)

{

int t = a[i];

a[i] = a[j];

a[j] = t;

}

## given m x n matrix print all the possible paths top to down.

Example

1 2 3

4 5 6

7 8 9

path for root(0,0) 1

1-4-7

1-4-8

1-5-7

1-5-8

1-5-9

similarly path for 2(0,1)

2-4-7

2-4-8

2-5-7

2-5-8

2-5-9

2-6-8

2-6-9

note- root 1 can go to middle down or right down since there is no left index available. if root element has left middle and right it can go to all those paths like 2 or 5.

follow up : provide the path which has maximum path sum.

code in java."

Code 1

public class MatrixPaths {

public static void printpaths(int[][] matrix) {

int[] path = new int[matrix.length];

for (int i = 0; i < matrix[0].length; i++)

printpaths(matrix, path, 0, 0, i);

}

private static void printpaths(int[][] matrix, int[] path, int index, int row, int column) {

path[index++] = matrix[row][column];

row++;

if (row == matrix.length)

print(path);

else if (row < matrix.length) {

for (int i = column - 1; i <= column + 1; i++)

if (i > -1 && i < matrix[0].length)

printpaths(matrix, path, index, row, i);

}

}

private static void print(int[] path) {

for (int i = 0; i < path.length; i++)

System.out.print(path[i] + " ");

System.out.println();

}

public static void main(String args[]) {

int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

printpaths(matrix);

}

}

Code 2

public class Main{

public static void main(String[] args){

int[][]A = {{1,2,3},{4,5,6},{7,8,9}};

printPath(A);

}

static void printPath(int[][]A){

printPath(A,0,0,"");

}

private static void printPath(int[][] A, int i, int j, String sofar) {

if(i<A.length){

if(sofar.length() == 0)

sofar += A[i][j];

else

sofar += "-" + A[i][j];

if(i==A.length-1){

printPath(A,i+1,j,sofar);

return;

}

//Handle 3 cases with j-1,j and j+1

if(j-1>=0)

printPath(A,i+1,j-1,sofar);

printPath(A,i+1,j,sofar);

if(j+1<A[0].length)

printPath(A,i+1,j+1,sofar);

}else{

System.out.println(sofar);

}

}

}

## I have an arrayList A which contains say 2,3,5,7,8. I have another arrayList B which contains 1, 3 Now taking the elements of B as the locations, I need to remove the elements of A present in that locations. So, basically I need to remove the element 2(position 1) and 5(position 3) from A. How to achieve it as we know that once one element got removed from an arrayList,the positions will be auto adjusted.

If B is sorted, then start removing the elements from end.

If B is not sorted, sort it first and then start removing elemetns from the end.

If B cannot be sorted, then mark the elements in A and in second pass remove the elements from A.

## You are given N points in a X-Y plane. Find the two closest points out of them.

We are given an array of n points in the plane, and the problem is to find out the closest pair of points in the array. This problem arises in a number of applications. For example, in air-traffic control, you may want to monitor planes that come too close together, since this may indicate a possible collision. Recall the following formula for distance between two points p and q.  
[dist_formula](http://www.geeksforgeeks.org/wp-content/uploads/dist_formula.png)

The Brute force solution is O(n^2), compute the distance between each pair and return the smallest. We can calculate the smallest distance in O(nLogn) time using Divide and Conquer strategy. In this post, a O(n x (Logn)^2) approach is discussed. We will be discussing a O(nLogn) approach in a separate post.

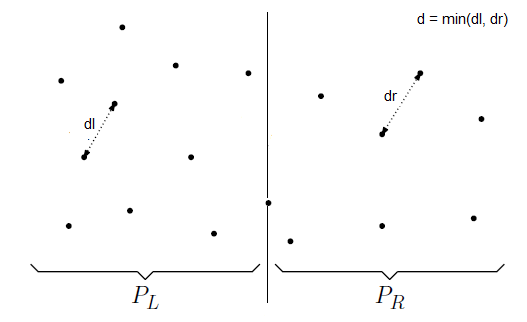
**Algorithm**  
Following are the detailed steps of a O(n (Logn)^2) algortihm.  
Input: An array of n points P[]  
Output: The smallest distance between two points in the given array.

As a pre-processing step, input array is sorted according to x coordinates.

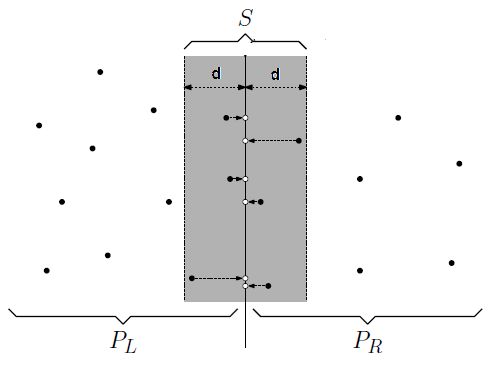
**1)** Find the middle point in the sorted array, we can take P[n/2] as middle point.

**2)** Divide the given array in two halves. The first subarray contains points from P[0] to P[n/2]. The second subarray contains points from P[n/2+1] to P[n-1].

**3)** Recursively find the smallest distances in both subarrays. Let the distances be dl and dr. Find the minimum of dl and dr. Let the minimum be d.

[](http://www.geeksforgeeks.org/wp-content/uploads/divide_points_new1.png)

**4)** From above 3 steps, we have an upper bound d of minimum distance. Now we need to consider the pairs such that one point in pair is from left half and other is from right half. Consider the vertical line passing through passing through P[n/2] and find all points whose x coordinate is closer than d to the middle vertical line. Build an array strip[] of all such points.

[](http://www.geeksforgeeks.org/wp-content/uploads/strip_closesr1.png)

**5)** Sort the array strip[] according to y coordinates. This step is O(nLogn). It can be optimized to O(n) by recursively sorting and merging.

**6)** Find the smallest distance in strip[]. This is tricky. From first look, it seems to be a O(n^2) step, but it is actually O(n). It can be proved geometrically that for every point in strip, we only need to check at most 7 points after it (note that strip is sorted according to Y coordinate). See [this](http://www.cs.umd.edu/class/fall2012/cmsc451/Lects/lect10.pdf) for proof.

**7)** Finally return the minimum of d and distance calculated in above step (step 6)

**Implementation**  
Following is C/C++ implementation of the above algorithm.

// A divide and conquer program in C/C++ to find the smallest distance from a

// given set of points.

#include <stdio.h>

#include <float.h>

#include <stdlib.h>

#include <math.h>

// A structure to represent a Point in 2D plane

struct Point

{

int x, y;

};

/\* Following two functions are needed for library function qsort().

Refer: http://www.cplusplus.com/reference/clibrary/cstdlib/qsort/ \*/

// Needed to sort array of points according to X coordinate

int compareX(const void\* a, const void\* b)

{

Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;

return (p1->x - p2->x);

}

// Needed to sort array of points according to Y coordinate

int compareY(const void\* a, const void\* b)

{

Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;

return (p1->y - p2->y);

}

// A utility function to find the distance between two points

float dist(Point p1, Point p2)

{

return sqrt( (p1.x - p2.x)\*(p1.x - p2.x) +

(p1.y - p2.y)\*(p1.y - p2.y)

);

}

// A Brute Force method to return the smallest distance between two points

// in P[] of size n

float bruteForce(Point P[], int n)

{

float min = FLT\_MAX;

for (int i = 0; i < n; ++i)

for (int j = i+1; j < n; ++j)

if (dist(P[i], P[j]) < min)

min = dist(P[i], P[j]);

return min;

}

// A utility function to find minimum of two float values

float min(float x, float y)

{

return (x < y)? x : y;

}

// A utility function to find the distance beween the closest points of

// strip of given size. All points in strip[] are sorted accordint to

// y coordinate. They all have an upper bound on minimum distance as d.

// Note that this method seems to be a O(n^2) method, but it's a O(n)

// method as the inner loop runs at most 6 times

float stripClosest(Point strip[], int size, float d)

{

float min = d; // Initialize the minimum distance as d

qsort(strip, size, sizeof(Point), compareY);

// Pick all points one by one and try the next points till the difference

// between y coordinates is smaller than d.

// This is a proven fact that this loop runs at most 6 times

for (int i = 0; i < size; ++i)

for (int j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)

if (dist(strip[i],strip[j]) < min)

min = dist(strip[i], strip[j]);

return min;

}

// A recursive function to find the smallest distance. The array P contains

// all points sorted according to x coordinate

float closestUtil(Point P[], int n)

{

// If there are 2 or 3 points, then use brute force

if (n <= 3)

return bruteForce(P, n);

// Find the middle point

int mid = n/2;

Point midPoint = P[mid];

// Consider the vertical line passing through the middle point

// calculate the smallest distance dl on left of middle point and

// dr on right side

float dl = closestUtil(P, mid);

float dr = closestUtil(P + mid, n-mid);

// Find the smaller of two distances

float d = min(dl, dr);

// Build an array strip[] that contains points close (closer than d)

// to the line passing through the middle point

Point strip[n];

int j = 0;

for (int i = 0; i < n; i++)

if (abs(P[i].x - midPoint.x) < d)

strip[j] = P[i], j++;

// Find the closest points in strip. Return the minimum of d and closest

// distance is strip[]

return min(d, stripClosest(strip, j, d) );

}

// The main functin that finds the smallest distance

// This method mainly uses closestUtil()

float closest(Point P[], int n)

{

qsort(P, n, sizeof(Point), compareX);

// Use recursive function closestUtil() to find the smallest distance

return closestUtil(P, n);

}

// Driver program to test above functions

int main()

{

Point P[] = {{2, 3}, {12, 30}, {40, 50}, {5, 1}, {12, 10}, {3, 4}};

int n = sizeof(P) / sizeof(P[0]);

printf("The smallest distance is %f ", closest(P, n));

return 0;

}

Output:

The smallest distance is 1.414214

**Time Complexity** Let Time complexity of above algorithm be T(n). Let us assume that we use a O(nLogn) sorting algorithm. The above algorithm divides all points in two sets and recursively calls for two sets. After dividing, it finds the strip in O(n) time, sorts the strip in O(nLogn) time and finally finds the closest points in strip in O(n) time. So T(n) can expressed as follows  
T(n) = 2T(n/2) + O(n) + O(nLogn) + O(n)  
T(n) = 2T(n/2) + O(nLogn)  
T(n) = T(n x Logn x Logn)

**Notes**  
**1)** Time complexity can be improved to O(nLogn) by optimizing step 5 of the above algorithm. We will soon be discussing the optimized solution in a separate post.  
**2)** The code finds smallest distance. It can be easily modified to find the points with smallest distance.  
**3)** The code uses quick sort which can be O(n^2) in worst case. To have the upper bound as O(n (Logn)^2), a O(nLogn) sorting algorithm like merge sort or heap sort can be used

## Given a few numbers, how will you push them into a stack such that whenever I pop the top most element from that stack, I get the minimum number from the bunch. Also he wanted me to tell the pop push algorithm such that the order of insertion and popping should be O(1).

Maintain two stacks. Push the elements entered by user in stack1. And in stack2 push the current minimum element.

Pop: Pop the elements from both stack and return the element from stack2.

## Given a set of positive integers S = { a1,a2,a3,...,an} find two subsets s1 and s2 of A such that S2 = S - S1 and difference of | sum(S1) - sum(S2) | is minimum.

For example if we have a set S={12,4,7,1,6,3,3 }then S1= {12,6} and S2={ 4,7,1,3,3} such that sum(S1) - sum(S2) = 0 . It is not necessary that two subsets will always have the same sum.

This is example of famous **balanced partitioning** problem. You need to basically find if any subset of set S sums up to N=(a1+a2+a3+...an)/2. If yes then other set will automatically have sum (a1+..+an)/2. However if that is not true then we will try if a subset has sum N-1, N-2,N-3 etc. Now the question is how to find if a subset of S sums up to some number. For this we use dynamic programming.   
P(i,j) is true if there is a subset from among first i elements that sum up to j. The recurrence relation will be   
P(i,j) = 1 if P(i-1,j) = 1 // there is a subset of (i-1) elements that sum up to j   
OR if P(i-1, j-a(i) ) = 1 // include the ith element in the subset

## Given an arry Arr[N] of integers and a function int func(int x) that takes an integer and returns either 0 or 1 (depending on some property of the integer). Give the most efficient algorithm to store the numbers in the array in such a way that all numbers that return 1 should come before all numbers that return 0, when called upon by the function func

This is what partition and stable\_partition functions in stl do.

template< class BidirIt, class UnaryPredicate >

BidirectionalIterator **partition**( BidirIt first, BidirIt last,

UnaryPredicate p );

template< class ForwardIt, class UnaryPredicate >

ForwardIt **partition**( ForwardIt first, ForwardIt last,

UnaryPredicate p );

template< class BidirIt, class UnaryPredicate >

BidirIt **stable\_partition**( BidirIt first, BidirIt last, UnaryPredicate p );

template<class BidirIt, class UnaryPredicate>

BidirIt **partition**(BidirIt first, BidirIt last, UnaryPredicate p)

{

while (1) {

while ((first != last) && p(\*first)) {

++first;

}

if (first == last--) break;

while ((first != last) && !p(\*last)) {

--last;

}

if (first == last) break;

std::swap(\*first++, \*last);

}

return first;

}

## Design an algorithm to sort an array whose first n-sqrt(n) elements are already sorted. What is the complexity of the algorithm.

1. Sort the remaining sqrt(n) elements and do a inplace merge. Complexity O( n+ sqrt(n)\*log(sqrt(n)) )
2. Do insertion of the remaining sqrt(n) elements in sorted part. Complexity O( n\*sqrt(n) )

## you have to do the queue operation when you have a single stack..for example:-suppose there is a telephone bill populating in a stack now you have to process each request in a fifo manner.you can perform only push pop operation and allocating extra memory is not allowed.

As we have only one stack, we need to resort to using recursion for the deque operation.

In effect we are still using extra stack but it is implicit.

void dequeue(Stack\* s, int\* data) {

//Remove from head, head will be modified to head->next, so passing &s

int r = pop(&s);

if(s == NULL) {

// We have reached the end of the list and r contains the first element entered in the stack

\*data = r;

return;

}

else {

dequeue(s, data);

push(&s, r);

}

}

## find missing numbers in given billion number.( numbers lie between 1-k)

1. XOR if only one number is missing
2. Use bitset to set bits for all the numbers, as they are found. And the 0 bits will tell you the missing numbers.

## Sort 10 GB file using 2 GB memory. and complexity

External sorting. Divide 10GB data into 5 parts of 2GB each. Sort these 5 parts.

Then Merge these parts. Open 5 streams and put 2GB/5 records of each sorted parts in memory and start merging them. When a parts in-memory records are processed read more records for that part.

## Given 2 Tree, find out if they r exactly same or not

Modified Inorder Traversal(node \*head1,node \*head2)

{

if(both head1 head2 are NULL)

return true;

if(one of head1 n head2 NULL)

return false;

if data(head1) == data(head2)

return inorder(head1>left,head2->left)&&inorder(head1->right,head2->right);

return false;

}

## Given a graph of price variation of a stock over a period of 12 months, return the ideal time to buy and sell, for maximization of profit. Time duration is not constraint. Time duration needn't be minimum it can even be maximum 12 motnhs

Consider that the prices are given as an array. We need to find the minimum and maximum elements from this array.

Traverse the array, keeping track of the maximum and minimum values and updating when a new value is found.

Divide and conquer

## Given a sequence of charecters , print consequtive sequence of charecters alone in a line, other wise print it in a new line.

eg: ABCXYZACCD

o/p :

ABC

XYZ

A

C

CD"

void increasestring2(char\* str)

{

char prev = 0;

while(\*str)

{

if(prev==0 || \*str==prev)

printf("%c", \*str);

else

printf("\n%c", \*str);

prev = \*str;

str++;

}

}

## How to find the depth of a tree. Time and Memory optimization is emphasized

int FindDepth(Node \*node)

{

int leftDepth, rightDepth;

if (node == NULL)

return 0;

else{

leftDepth = FindDepth(node->left);

rightDepth = FindDepth(node->right);

if (leftDepth > rightDepth)

return (leftDepth+1);

else

return (rightDepth+1);

}

}

## What are the various ways to swap 2 variables

Answer :

a) Using temporary Variable

b) Usnig some Arithmentic operation

c) Using bitwise XOR operation

Which operation is better and Why ?

## Given a file, return the Top 5 frequently occuring list of words

## Design a Chess Game

## Design Snake and Ladder Game

## Given a white background with a black random shape scattering around. Describe and code an algorithm to count the number of black shapes. You can assume input to be an array of 3xMxN in RGB color space.

## Given one unsroted integer array, find out all the unique element in the array.

eg: Input: {23,53,1,3,6,23,1,7,9,53,9} Ouput;{3,6,7}

My solution:

Sort the array. Time: O(NlogN)

HashMap: Time: O(2N) Space: O(N)

Any improvement for this question?? Thanks."

## Given an integer n , you have to print all the ways in which n can be represented as sum of positive integers

N 2 positions x and n-x

## write a program to display the no. of 1,2,3,4... lettered words in paragraph.

example:

1.search

2.engine

3.needs

4.to

5.fast

//output

search engine needs to be fast.

asked in interview ,though i didn't understand the purpose.

## Count the number of unique element in array of numbers in minimum time complexity.

## Given an array of length N. How will you find the minimum length contiguous subarray whose sum is S and whose product is P . HereS and P will be given to you.

## Implement a search engine. The input will be 100 text files. Return the paragraph and the file in which the keyword appears.

## Implement a phone directory

## Implement Twitter's trending topic. The input is a text file. Implement it in C.

## Given a stack S, write a C program to sort the stack (in ascending order). You are not allowed to make any assumptions about how the stack is implemented; the only functions allowed to be used are: Push, Pop, Top, IsEmpty, IsFull.

We will have to use an auxiliary stack. Let’s say we want to sort the stack in ascending order, ie when you pop the element after sorting we will get the smallest element.

While(!S.IsEmpty())

s = Pop element from S

if top(S1) <= s

S1.push(s)

Else

While(top(S1) > s)

s1=S1.pop()

S.push(s1);

S1.push(s);

While(!S1.isEmpty())

S.push(S1.pop());

## There is a sequence of increasing numbers that have the same number of binary 1s in them. Given n, the number of 1 bits set in each number, write an algorithm or C program to find the n’th number in the series.

(1 <<( n+1)) – 3 Or (2<<n) - 3

01000

1<<n

00011

00101

00110

01010

01100

10100

11000

00111

01011

01101

01110

## You are given have a datatype, say X in C. Determine the size of the datatype, without declaring a variable or a pointer variable of that type, and, of course without using the sizeof operator.

#define mysizeof(X) ((char \*)((X \*)0 + 1) - (char \*)((X \*)0))

## Write a C Program to reverse a stack in place using recursion. You can only use the following ADT functions on stack: IsEmpty, IsFull, Push, Pop, Top. you can not use extra stack or any other data structure

The idea of the solution is to hold all values in Function Call Stack until the stack becomes empty. When the stack becomes empty, insert all held items one by one at the bottom of the stack.

For example, let the input stack be

1 <-- top

2

3

4

First 4 is inserted at the bottom.

4 <-- top

Then 3 is inserted at the bottom

4 <-- top

3

Then 2 is inserted at the bottom

4 <-- top

3

2

Then 1 is inserted at the bottom

4 <-- top

3

2

1

So we need a function that inserts at the bottom of a stack using the above given basic stack function. **//Below is a recursive function that inserts an element at the bottom of a stack.**

|  |
| --- |
| void insertAtBottom(struct sNode\*\* top\_ref, int item)  {     int temp;     if(isEmpty(\*top\_ref))     {         push(top\_ref, item);     }     else     {         /\* Hold all items in Function Call Stack until we reach end of         the stack. When the stack becomes empty, the isEmpty(\*top\_ref)         becomes true, the above if part is executed and the item is         inserted at the bottom \*/       temp = pop(top\_ref);       insertAtBottom(top\_ref, item);         /\* Once the item is inserted at the bottom, push all the            items held in Function Call Stack \*/       push(top\_ref, temp);     }  } |

**//Below is the function that reverses the given stack using insertAtBottom()**

|  |
| --- |
| void reverse(struct sNode\*\* top\_ref)  {    int temp;    if(!isEmpty(\*top\_ref))    {        /\* Hold all items in Function Call Stack until we reach end of       the stack \*/      temp = pop(top\_ref);      reverse(top\_ref);        /\* Insert all the items (held in Function Call Stack) one by one         from the bottom to top. Every item is inserted at the bottom \*/      insertAtBottom(top\_ref, temp);    }  } |

## Write a program to find and print the 1500’th ugly number.

**METHOD 1 (Simple)**  
**Algorithm:**  
Loop for all positive integers until ugly number count is smaller than n, if an integer is ugly than increment ugly number count.

To check if a number is ugly, divide the number by greatest divisible powers of 2, 3 and 5, if the number becomes 1 then it is an ugly number otherwise not.

For example, let us see how to check for 300 is ugly or not. Greatest divisible power of 2 is 4, after dividing 300 by 4 we get 75. Greatest divisible power of 3 is 3, after dividing 75 by 3 we get 25. Greatest divisible power of 5 is 25, after dividing 25 by 25 we get 1. Since we get 1 finally, 300 is ugly number.

**Implementation:**

|  |
| --- |
| # include<stdio.h>  # include<stdlib.h>    /\*This function divides a by greatest divisible    power of b\*/  int maxDivide(int a, int b)  {    while (a%b == 0)     a = a/b;    return a;  }    /\* Function to check if a number is ugly or not \*/  int isUgly(int no)  {    no = maxDivide(no, 2);    no = maxDivide(no, 3);    no = maxDivide(no, 5);      return (no == 1)? 1 : 0;  }    /\* Function to get the nth ugly number\*/  int getNthUglyNo(int n)  {    int i = 1;    int count = 1;   /\* ugly number count \*/      /\*Check for all integers untill ugly count      becomes n\*/    while (n > count)    {      i++;      if (isUgly(i))        count++;    }    return i;  }    /\* Driver program to test above functions \*/  int main()  {      unsigned no = getNthUglyNo(150);      printf("150th ugly no. is %d ",  no);      getchar();      return 0;  } |

This method is not time efficient as it checks for all integers until ugly number count becomes n, but space complexity of this method is O(1)

**METHOD 2 (Use Dynamic Programming)**

Here is a time efficient solution with O(n) extra space

**Algorithm:**

1 Declare an array for ugly numbers: ugly[150]

2 Initialize first ugly no: ugly[0] = 1

3 Initialize three array index variables i2, i3, i5 to point to

1st element of the ugly array:

i2 = i3 = i5 =0;

4 Initialize 3 choices for the next ugly no:

next\_mulitple\_of\_2 = ugly[i2]\*2;

next\_mulitple\_of\_3 = ugly[i3]\*3

next\_mulitple\_of\_5 = ugly[i5]\*5;

5 Now go in a loop to fill all ugly numbers till 150:

For (i = 1; i < 150; i++ )

{

/\* These small steps are not optimized for good

readability. Will optimize them in C program \*/

next\_ugly\_no = Min(next\_mulitple\_of\_2,

next\_mulitple\_of\_3,

next\_mulitple\_of\_5);

if (next\_ugly\_no == next\_mulitple\_of\_2)

{

i2 = i2 + 1;

next\_mulitple\_of\_2 = ugly[i2]\*2;

}

if (next\_ugly\_no == next\_mulitple\_of\_3)

{

i3 = i3 + 1;

next\_mulitple\_of\_3 = ugly[i3]\*3;

}

if (next\_ugly\_no == next\_mulitple\_of\_5)

{

i5 = i5 + 1;

next\_mulitple\_of\_5 = ugly[i5]\*5;

}

ugly[i] = next\_ugly\_no

}/\* end of for loop \*/

6.return next\_ugly\_no

**Example:**  
Let us see how it works

initialize

ugly[] = | 1 |

i2 = i3 = i5 = 0;

First iteration

ugly[1] = Min(ugly[i2]\*2, ugly[i3]\*3, ugly[i5]\*5)

= Min(2, 3, 5)

= 2

ugly[] = | 1 | 2 |

i2 = 1, i3 = i5 = 0 (i2 got incremented )

Second iteration

ugly[2] = Min(ugly[i2]\*2, ugly[i3]\*3, ugly[i5]\*5)

= Min(4, 3, 5)

= 3

ugly[] = | 1 | 2 | 3 |

i2 = 1, i3 = 1, i5 = 0 (i3 got incremented )

Third iteration

ugly[3] = Min(ugly[i2]\*2, ugly[i3]\*3, ugly[i5]\*5)

= Min(4, 6, 5)

= 4

ugly[] = | 1 | 2 | 3 | 4 |

i2 = 2, i3 = 1, i5 = 0 (i2 got incremented )

Fourth iteration

ugly[4] = Min(ugly[i2]\*2, ugly[i3]\*3, ugly[i5]\*5)

= Min(6, 6, 5)

= 5

ugly[] = | 1 | 2 | 3 | 4 | 5 |

i2 = 2, i3 = 1, i5 = 1 (i5 got incremented )

Fifth iteration

ugly[4] = Min(ugly[i2]\*2, ugly[i3]\*3, ugly[i5]\*5)

= Min(6, 6, 10)

= 6

ugly[] = | 1 | 2 | 3 | 4 | 5 | 6 |

i2 = 3, i3 = 2, i5 = 1 (i2 and i3 got incremented )

Will continue same way till I < 150

**Program:**

|  |
| --- |
| # include<stdio.h>  # include<stdlib.h>  # define bool int    /\* Function to find minimum of 3 numbers \*/  unsigned min(unsigned , unsigned , unsigned );    /\* Function to get the nth ugly number\*/  unsigned getNthUglyNo(unsigned n)  {      unsigned \*ugly =               (unsigned \*)(malloc (sizeof(unsigned)\*n));      unsigned i2 = 0, i3 = 0, i5 = 0;      unsigned i;      unsigned next\_multiple\_of\_2 = 2;      unsigned next\_multiple\_of\_3 = 3;      unsigned next\_multiple\_of\_5 = 5;      unsigned next\_ugly\_no = 1;      \*(ugly+0) = 1;        for(i=1; i<n; i++)      {         next\_ugly\_no = min(next\_multiple\_of\_2,                             next\_multiple\_of\_3,                             next\_multiple\_of\_5);         \*(ugly+i) = next\_ugly\_no;         if(next\_ugly\_no == next\_multiple\_of\_2)         {             i2 = i2+1;             next\_multiple\_of\_2 = \*(ugly+i2)\*2;         }         if(next\_ugly\_no == next\_multiple\_of\_3)         {             i3 = i3+1;             next\_multiple\_of\_3 = \*(ugly+i3)\*3;         }         if(next\_ugly\_no == next\_multiple\_of\_5)         {             i5 = i5+1;             next\_multiple\_of\_5 = \*(ugly+i5)\*5;         }      } /\*End of for loop (i=1; i<n; i++) \*/      return next\_ugly\_no;  }    /\* Function to find minimum of 3 numbers \*/  unsigned min(unsigned a, unsigned b, unsigned c)  {      if(a <= b)      {        if(a <= c)          return a;        else          return c;      }      if(b <= c)        return b;      else        return c;  }    /\* Driver program to test above functions \*/  int main()  {      unsigned no = getNthUglyNo(150);      printf("%dth ugly no. is %d ", 150, no);      getchar();      return 0;  } |

**Algorithmic Paradigm:** Dynamic Programming  
**Time Complexity:** O(n)  
**Storage Complexity:** O(n)

## Write a method to generate a random number between 1 and 7, given a method that generates a random number between 1 and 5 (i.e., implement rand7() using rand5()).

Rand7()

{

N=0;

For(int I = 0; i< 3 ; i++) {

N = Rand5() %2;

N<<1

}

Return N;

}

## Given two sets, write a function to provide the union of them.

## You are given a list of Ball objects. Each Ball is either Red or Blue. Write a function that partitions these balls so that all of the balls of each color are contiguous. Return the index of the first ball of the second color (your result can be Red balls, then Blue balls, or the other way around). In haskell, you’ll probably want to return a ([Ball],Int)."

Take to index. I and j

Loop: I++ if ball is red

Loop: j—if ball is blue

If i<j swap I , j

Else

Return i

## Given a linked list, split it into two lists - one for the front half and, one for the last half. If odd number of elements are present the extra node should go to the front list

slow=fast=head;

while(fast->next!=NULL && fast->next->next!=NULL)

{

slow=slow->next;

fast=fast->next->next;

}

front=head;

if(slow->next!=NULL)

{

last=slow->next;

}

else

{

last=NULL;

}

## Remove duplicates from a linked list

/\* Function to remove duplicates from a unsorted linked list \*/

void removeDuplicates(struct node \*start)

{

  struct node \*ptr1, \*ptr2, \*dup;

  ptr1 = start;

  /\* Pick elements one by one \*/

  while(ptr1 != NULL && ptr1->next != NULL)

  {

     ptr2 = ptr1;

     /\* Compare the picked element with rest of the elements \*/

     while(ptr2->next != NULL)

     {

       /\* If duplicate then delete it \*/

       if(ptr1->data == ptr2->next->data)

       {

          /\* sequence of steps is important here \*/

          dup = ptr2->next;

          ptr2->next = ptr2->next->next;

          free(dup);

       }

       else /\* This is tricky \*/

       {

          ptr2 = ptr2->next;

       }

     }

     ptr1 = ptr1->next;

  }

}

**METHOD 2 (Use Sorting)**  
In general, Merge Sort is the best suited sorting algorithm for sorting linked lists efficiently.  
1) Sort the elements using Merge Sort. We will soon be writing a post about sorting a linked list. O(nLogn)  
2) Remove duplicates in linear time using the [algorithm for removing duplicates in sorted Linked List. O(n)](http://geeksforgeeks.org/?p=5075)

**METHOD 3 (Use Hashing)**  
We traverse the link list from head to end. For every newly encountered element, we check whether it is in the hash table: if yes, we remove it; otherwise we put it in the hash table.

Thanks to bearwang for suggesting this method.

Time Complexity: O(n) on average (assuming that hash table access time is O(1) on average).

## You are given: 3 types of vehicles: Motorbike, Car, and a special type of car for the handicapped. 3 types of parking: Motorbike parking, Car parking, handicapped car parking. Motorbikes and cars can only park in their designated parkings, while the handicapped cars can park either in their own parking or the regular car parking. How would you model this as classes? Explain your methods.

Vehicle Classes: MotorBike, Car, HandicapCar

Parking Classes: MoterBikeParking, CarParking, HandicapParking

MotorBike will always get MotorBikeParking if available, else null

Car will get CarParking, if av else null

HandicapCar can get HandicapParking or CarParking, if available,else null

Need a factory class to create instances based on the available parking slots and type of vehicle being parked.

## Find the missing element in a sorted array in most optimum running time (O(log n))

Let say the range is (m, n). There are n – m numbers in the array and one is missing.

Ex (1, 11) , 10 elements and one is missing

At index I the number should be i+m. if it is then search in right part else search in left part

## Given a balanced BST tree, write a function to replace the root with a node that belongs to the original tree.

## Given n non overlapping intervals and an element. Write a program to find the interval into which this element falls.

If the intervals are not sorted then this can be done in O(n) time

If sorted on start-point of intervals,

## Design a stack which supports push, pop, min and max operations in O(1).

Use two extra stacks to keep track of min and max at each level. The minStack will hold the minimum element till now in our main stack and maxStack will hold the maximum element in our main stack.

Each push/pop operation will update all the tree stacks.

## There is a blank disc. You are given two colors of paint (black and white) . A sensor can recognize the color painted on the disc and produce an output. Paint the disc in a way such that you can find the direction of rotation by looking at the output (BWBWBWBW... etc). Find the minimum number of sectors you will need to paint"

B W B B W W

## An array of size n, has n/2 unique elements and n/2 occurences of an element. Find the non-unique element in linear time?

Same as 1.37

int find\_majority(int \*a, int n)

{

int count = 1;

int currentIndex = 0;

int i;

for (i = 1 ; i < n; i ++)

{

if (a[i] == a[currentIndex])

count++;

else

count--;

if (count == 0)

{

currentIndex = i;

count = 1;

}

}

return a[currentIndex];

}

## You are provided with a stream of numbers, design a data structure to store the numbers in the stream along with their no. of occurrences.

Map <int,int> num\_stream;

Hash\_map;

## Count number of lines, characters and words in a file (Given that we don’t have much access to flashy java methods like readline, String methods like indexOf etc.)

Chars: keep on incrementing the count at every char, which is not newline, tab

Word termination at : space, tab, newline, period, comma, semicolon etc

Line termination at: newline

## Write a function which takes as parameters one regular expression (only ? and \* are the special characters) and a string and returns whether the string matched the regular expression.

bool MatchReg(string regexp, string toMatch)

{

}

## "There are four dogs, each at the counter of a large square. Each of the dogs begins chasing the dog clockwise from it. All of the dogs run at the same speed. All continously adjust their direction so that they are always heading straight towards their clockwise neighbor. How long does it take for the dogs to catch each other? Where does this happen? (Hint: Dog’s are moving in a symmetrical fashion, not along the edges of the square)."

They meet at the center of the square and move on a circular trajectory. The distance from a corner to the center is therefore 1/4 of the circumference of the circle (2\*pi\*radius). The radius is 1/2 the length of a side of the square, L. So the distance traveled by a dog is:   
1/4 \* 2 \* pi \* 1/2 \* L = pi\*L/4   
Since time = distance/velocity,   
t = (pi\*L)/(4\*v)

## "Three strings say A,B,C are given to you. Check weather 3rd string is interleaved from string A and B.

Ex: A=""abcd"" B=""xyz"" C=""axybczd"". answer is yes."

Take three indexers i1 i2 i3

if C[i3] == A[i1]

i3++; i1++;

else if C[i3] == **B[i2]**

**i3++; i2++**

**else**

**NO;**

## Given that you can take one step or two steps forward from a given step. So find the total number of ways of reaching Nth step.

P(N) = P(N-1) + P(N-2)

## Count the number of set bits in a number. Now optimize for speed. Now optimize for size

int BitCount (unsigned int u)

{

unsigned int uCount=0 ;

for(; u; u&=(u-1))

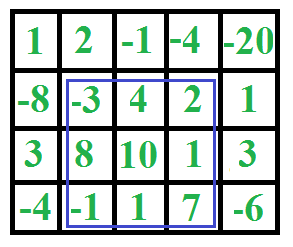
uCount++;

return uCount ;

}

## Suppose you have an NxN matrix of positive and negative integers. Write some code that finds the sub-matrix with the maximum sum of its elements.

Given a 2D array, find the maximum sum subarray in it. For example, in the following 2D array, the maximum sum subarray is highlighted with blue rectangle and sum of this subarray is 29.

[](http://www.geeksforgeeks.org/wp-content/uploads/rectangle.png)

This problem is mainly an extension of [Largest Sum Contiguous Subarray for 1D array](http://www.geeksforgeeks.org/largest-sum-contiguous-subarray/).

The **naive solution** for this problem is to check every possible rectangle in given 2D array. This solution requires 4 nested loops and time complexity of this solution would be O(n^4).

**Kadane’s algorithm** for 1D array can be used to reduce the time complexity to O(n^3). The idea is to fix the left and right columns one by one and find the maximum sum contiguous rows for every left and right column pair. We basically find top and bottom row numbers (which have maximum sum) for every fixed left and right column pair. To find the top and bottom row numbers, calculate sun of elements in every row from left to right and store these sums in an array say temp[]. So temp[i] indicates sum of elements from left to right in row i. If we apply Kadane’s 1D algorithm on temp[], and get the maximum sum subarray of temp, this maximum sum would be the maximum possible sum with left and right as boundary columns. To get the overall maximum sum, we compare this sum with the maximum sum so far.

// Program to find maximum sum subarray in a given 2D array

#include <stdio.h>

#include <string.h>

#include <limits.h>

#define ROW 4

#define COL 5

// Implementation of Kadane's algorithm for 1D array. The function returns the

// maximum sum and stores starting and ending indexes of the maximum sum subarray

// at addresses pointed by start and finish pointers respectively.

int kadane(int\* arr, int\* start, int\* finish, int n)

{

// initialize sum, maxSum and start

int sum = 0, maxSum = INT\_MIN, i;

// needed if sum NEVER becomes less than 0

\*start = 0;

// Standard Kadane's algorithm. See following link

// http://www.geeksforgeeks.org/largest-sum-contiguous-subarray/

for (i = 0; i < n; ++i)

{

sum += arr[i];

if (sum < 0)

{

sum = 0;

\*start = i+1;

}

else if (sum > maxSum)

{

maxSum = sum;

\*finish = i;

}

}

return maxSum;

}

// The main function that finds maximum sum rectangle in M[][]

void findMaxSum(int M[][COL])

{

// Variables to store the final output

int maxSum = 0, finalLeft, finalRight, finalTop, finalBottom;

int left, right, i;

int temp[ROW], sum, start, finish;

// Set the left column

for (left = 0; left < COL; ++left)

{

// Initialize all elements of temp as 0

memset(temp, 0, sizeof(temp));

// Set the right column for the left column set by outer loop

for (right = left; right < COL; ++right)

{

// Calculate sum between current left and right for every row 'i'

for (i = 0; i < ROW; ++i)

temp[i] += M[i][right];

// Find the maximum sum subarray in temp[]. The kadane() function

// also sets values of start and finish. So 'sum' is sum of

// rectangle between (start, left) and (finish, right) which is the

// maximum sum with boundary columns strictly as left and right.

sum = kadane(temp, &start, &finish, ROW);

// Compare sum with maximum sum so far. If sum is more, then update

// maxSum and other output values

if (sum > maxSum)

{

maxSum = sum;

finalLeft = left;

finalRight = right;

finalTop = start;

finalBottom = finish;

}

}

}

// Print final values

printf("(Top, Left) (%d, %d)\n", finalTop, finalLeft);

printf("(Bottom, Right) (%d, %d)\n", finalBottom, finalRight);

printf("Max sum is: %d\n", maxSum);

}

// Driver program to test above functions

int main()

{

int M[ROW][COL] = {{1, 2, -1, -4, -20},

{-8, -3, 4, 2, 1},

{3, 8, 10, 1, 3},

{-4, -1, 1, 7, -6}

};

findMaxSum(M);

return 0;

}

Output:

(Top, Left) (1, 1)

(Bottom, Right) (3, 3)

Max sum is: 29

Time Complexity: O(n^3)

## Given the sequence S1 = {a,b,c,d,...,x,y,z,aa,ab,ac.... } and given that this sequence corresponds (term for term) to the sequence S2 = {1,2,3,4,....} Write code to convert an element of S1 to the corresponding element of S2. Write code to convert an element of S2 to the corresponding element of S1.

#include<stdio.h>

#include<string.h>

#include<malloc.h>

#include<math.h>

// Utility Function

char\* reverseString(char\* str)

{

int start = 0;

int end = getLength(str) - 1;

while(start < end)

{

char ch = str[end];

str[end] = str[start];

str[start] = ch;

start++; end--;

}

return str;

}

// Utility Functions

int getLength(char\* str)

{

int i = 0;

while(\*str)

{

i++;

str++;

}

return i;

}

//Works only for lower case characters a-z

int getEquivalentNumber(char\* input)

{

int code = 0;

int length = getLength(input);

--length;

while(length >= 0)

{

int value = (\*input) - 96;

code = code + value\*(pow(26,length));

length--;

input++;

}

return code;

}

//Works only for lower case characters a-z

char\* getEquivalentCode(int input)

{

char\* result = (char\*)malloc(10\*sizeof(char));

char\* tempResult = result;

int alphabet= 26;

while(input)

{

int ch = input % alphabet;

input = (input - ch) / alphabet;

\*result = (ch + 96);

result++;

}

\*result = '\0';

return reverseString(tempResult);

}

int main()

{

char input[5] = "kds";

int result = getEquivalentNumber(input);

printf("%s equivalent to %d\n", input,result);

int \_input = 7556;

char\* \_result = getEquivalentCode(\_input);

printf("%d equivalent to %s", \_input,\_result);

}

## Given N computers networked together, with each computer storing N integers, describe a procedure for finding the median of all of the numbers. Assume that a computer can only hold O(N) integers (i.e. no computer can store all N^2 integers). Also assume that there exists a computer on the network without integers, that we can use to interface with the computers storing the integers.

Each machine needs to be able to   
(a) Sort the numbers:

Sort();

(b) Given a value x, return the number of elements < x, == x, > x.

int LessThanCount(int x);

int EqualCount(int x);

int BiggerThanCount(int x);

(c) Given indexes begin and last, return a median of the values between begin & last (inclusive).

int Median(int begin, int last);

All 3 need additional constant space which should be OK.   
  
Now, the master machine will keep values begin[i] and last[i] for each machine (and 3 more for temporary storage). (2+3)\*N = O(n) data.   
  
Algorithm:

(0) Initially, set begin[i] = 0, last[i] = N-1 for all i, 0 <= i < N.

(1) Sort the data on each machine. Sort();

Repeat:

(2) Find a machine k for which last[k]-begin[k] is maximum

(3) Ask machine k for median, med = Median(begin[k], last[k]);

(4) For each machine i, get 3 additional numbers:

smaller[i] = LessThanCount(med);

equal[i] = EqualCount(med);

bigger[i] = BiggerCount(med);

(5) Sum all these numbers to obtain

TotalSmaller, TotalEqual, and TotalBigger

(6) if (abs(TotalSmaller-TotalBigger) <= TotalEqual)

We are done, return x.

(7) if not, we have 2 choices:

(a) TotalSmaller + TotalEqual < TotalBigger (i.e. x is too small)

Set begin[i] = smaller[i] + equal[i].

(b) TotalSmaller > TotalEqual + TotalBigger (i.e. x is too big)

Set last[i] = smaller[i] - 1.

Note that at each step, we are either increasing begin[i] or decreasing last[i] and therefore the pool of possible values is being reduced. This is because the new x is picked from values that are bigger than x\_s and smaller than x\_b, where x\_s is the most recent x that was too small and x\_b is the most recent x that was too big.   
  
The biggest concern here is obviously the space and there we used O(N) per machine as required.   
  
The running time is something like   
O(N\*log\_N) for sorting in paralel on all machines (Heapsort, no extra space)   
+   
O(N\*log\_N) for partitioning   
-- We have total of O(log\_n) iterations, at least on average. In each iteration, each machine will take at most O(n) time to return smaller[i], equal[i], and bigger[i] in parallel, plus we need to add these and assign new begin/last which also is O(n).   
  
Total = O(N\*log\_N) time, O(N) space per machine.

## "A man has two paper cubes on his desk. Every day he arranges both cubes so that the front faces show the current day of the month. What numbers are required on the faces of the cubes to allow this for all possible days in the calendar?"

012345   
012678

## A band is going in the street with a constant speed. Someone in the last row has a dog. The dog runs ahead, reaches the front row of the band and gets back to it's owner. The dog's speed was constant all the way and while it was running the band passed 50 feet. Find the length of the dog's path,if the distance between the front and the rear row of the band is 50 feet.

## There are 100 doors in a row that are all initially closed. You make 100 passes by the doors starting with the first door every time. The first time through you visit every door and toggle the door (if the door is closed, you open it, if its open, you close it). The second time you only visit every 2nd door (door #2, #4, #6). the third time, every 3rd door (door #3, #6, #9), etc, until you only visit the 100th door. What is the state of each door after the last pass?

## An apple is in the shape of a ball of radius 31 mm. A worm gets into the apple and digs a tunnel of total length 61 mm, and then leaves the apple. (The tunnel need not be a straight line.) Prove that one can cut the apple with a straight slice through the center so that one of the two halves is not rotten.

As the tunnel is of length 61mm, so it is definitely not going through the center. So its possible to cut the apple in half with complete tunnel in one half.

## You have a machine which can do only multiply by 2, divide by 2 and Addition of 2 numbers. Write a detailed algorithm to multiply any two numbers, in this kind of a machine.

int Multiply(int a, int b)

{

if (a == 0 || b == 0) return 0;

int result = 0;

while (b > 0)

{

if (b % 2 != 0) // the same as if (b != (b / 2) \* 2)

result = result + a;

a \*= 2;

b /= 2;

}

return result;

}

## An array is of size N with integers between 0 and 1024(repetitions allowed). Another array of integers is of size M with no constraints on the numbers. Find which elements of first array are present in the second array. (If you are using extra memory, think of minimizing that still, using bitwise operators)

0 to 1024 numbers can fit in 210 bits. Ie 27 bytes ie 25 ints. So define a int bitset[32]. Set the bits for number present in first array and check for numbers from second array if the bit is set for it.

## How many matches will be played in a knockout tournament between 9 teams get the general formula for n teams?

8

For n it will be n/2+n/4+n/8 … 1 + (1 more if n is odd)

## design a datastructure to represent the movement of a knight on a chess board

## Write an algorithm to traverse a knight covering all the squares on a chessboard starting at a particular point.

## "There is a temple, whose premises have a garden and a pond. It has 4 idols, each of Ram, Shiv, Vishnu and Durga. The priest plucks x flowers from the garden and places them in the pond. The number of flowers doubles up, and he picks y flowers out of them and goes to offer it to Lord Ram. By the time he reaches to the pond, he finds the remaining flowers also have doubled up in the meantime, so he again picks up y from the pond and goes to Lord Shiv.This process is repeated till all the Gods have y flowers offered to them, such that in the end no flower is left in the pond. Find x and y."

2x-y

4x-3y

8x-7y

16x-15y=0 => x=15, y=16

## Given that you have one string of length N and M small strings of length L . How do you efficiently find the occurrence of each small string in the larger one ?

Suffix tree

## You are given with three sorted arrays ( in ascending order), you are required to find a triplet ( one element from each array) such that distance is minimum. Distance is defined like this : If a[i], b[j] and c[k] are three elements then distance=max(abs(a[i]-b[j]),abs(a[i]-c[k]),abs(b[j]-c[k]))"Please give a solution in O(n) time complexity

We will store the min\_dist value such that min\_dist = abs(a[i] – b[j]) + abs(a[i] – c[k]) + abs(b[j] – c[k]), and also we should store these indexes i,j,k. So, by moving trough the sorted arrays using three pointers, we increment the pointer which element is minimum at each step, then check the distance and update min\_dist and indexes if needed.

## "There are a set of 'n' integers. Describe an algorithm to find for each of all its subsets of n-1 integers the product of its integers. For example, let consider (6, 3, 1, 2). We need to find these products : 6 \* 3 \* 1 = 18 6 \* 3 \* 2 = 36 3 \* 1 \* 2 = 6 6 \* 1 \* 2 = 12"

define a static variable that holds initially the product of all the numbers.   
den traverse thru the list of number and make the set of (product/a[i]) where 'a' is array of numbers of list of numbers and 'i' is the index while travellin thru the list.

## How would you determine if someone has won a game of tic-tac-toe on a board of any size?

## "Given two sequences of items, find the items whose absolute number increases or decreases the most when comparing one sequence with the other by reading the sequence only once."

## "How many different binary trees and binary search trees can be made from three nodes that contain the key values 1, 2 & 3?"

## "Given an expression tree with no parentheses in it, write the program to give equivalent infix expression with parenthesesinserted where necessary"

## "Given ships travel between points A and B, one every hour leaving from both ends (simultaneously), how many ships are required (minimum), if the journey takes 1hr 40 mts. How many ships does each ship encounter in its journey, and at what times?

Ans 4, 3 at 20 mts, 50 mts and 80 mts."

## Count the number of set bits in a number without using a loop.

## "How would you reverse the bits of a number with log N arithmetic operations, where N is the number of bits in the integer (eg 32,64..)"

## Delete a node from a binary tree and balance it. Write code for the former and explain the latter.

## "Given a maze with cheese at one place and a mouse at some entrance, write a program to direct the mouse to cheese correctly. (Assume there is a path). Following primitives are given: moveforward, turnright, turnleft, iswall?,ischeese?, eatcheese."

## "A car has speed of 72 64 56 in downhill, plain and uphill respectively . A guy travels in the car from Pt. A to pt. B in 4 Hrs and pt. B to pt. A in 4 Hrs and 40 min. what is the distance between A and B?"

## "Write a program to print the elements of a very long linked list in ascending order. There may be duplicates in the list. You cannot modify the list or create another one. Memory is tight, speed is not a problem."

## "A real life problem - A square picture is cut into 16 squares and they are shuffled. Write a program to rearrange the 16 squares to get the original big square"

## Given an array in which elements are unsorted. Write an algorithm that gives two indices n1,n2 such that if you sort just the elements of the array from n1 to n2, then the whole array will be sorted."

## given a word,convert it into a pallindrome with minimum addition of letters to it.letters can be added anywhere in the word.for eg if yahoo is given result shud be yahoohay.give a optimize soln

## What is Spring IOC?

## What is a deadlock and what are some of the ways to avoid a deadlock?

## Explain hashcode() and equals() methods? When would you override these methods? What does the hashcode() method in the Object class do?

## What is an immutable object? How do you implement one in Java? How to construct an immutable object that has an array/List as one of the instance properties?

## "Find the maximum subsequence sum of an array of integers which contains both positive and negative numbers and return the starting and ending indices within the array.

For example:

int array[] = {1, -2, -3, 4, 5, 7, -6}

The max subsquence sum is 4+5+7= 16 and start index is at 3 and end index is at 5."

## How can we traverse through all the files in a folder and the subfolders. What system calls should be used(in C).

## What is the difference between functors, call back functions and function pointers?

## Write a function to add an array of numbers.

## What's the difference between assignment operator and copy constructor

## What's the difference between pointer and reference

## What are call back functions?

## What is a functor?

## Write a unix program to count the number of lines in a text file

## What is pi?

## What is polymorphism

## What is the importance of the keyword static in java

## "Write a program to replace string 'us' with 'them' from the following String . Do not replace 'Us' as well as any string containing us $\_="Us? It usually rains when bus comes to us";"

## What is reflection

## Given an integer, print the closest number to it that is a palindrome - eg, the number "1224" would return "1221".

## How to develop a sorted lexicographic tree.

## How to find distance between two lines in a 3D plane

## Put eight chess queens on an 8×8 chessboard such that none of them is able to capture any other using the standard chess queen's moves

## "What data structure would you use to store distances between all the planets in a galaxy. (So there could be like a billion planets) Also steps in connecting a thin mobile client to connect to the server and get distances between one given planet and all the other planets in that galaxy."

## "There are given n men and n women. Each woman ranks all men in order of her preference (her first choice, her second choice, and so on). Similarly, each man sorts all women according to his preference. The goal is to arrange n marriages in such a way that if a man m prefers some woman w more than his wife, and w prefers m more then her husband a new marriage occurs between w and m. If w prefers her husband more, then she stays married to him. This problem always has a solution and your task is to find one."

## "Johnny was asked by his math teacher to compute nn (n to the power of n, where n is an integer), and has to read his answer out loud. This is a bit of a tiring task, since the result is probably an extremely large number, and would certainly keep Johnny occupied for a while if he were to do it honestly. But Johnny knows that the teacher will certainly get bored when listening to his answer, and will sleep through most of it! So, Johnny feels he will get away with reading only the first k digits of the result before the teacher falls asleep, and then the last k digits when the teacher wakes up. Write a program to help Johnny to compute the digits he will need to read out."

## "The Chef has one long loaf of bread. Let us say, of length 1. He wants to cut it into as many little loaves as he can. But he wants to adhere to the following rule: At any moment, the length of the longest loaf which he possesses may not be larger than the length of shortest one, times some constant factor. Every time, he is only allowed to cut exactly one loaf into two shorter ones.

Input

One floating-point number, 1 k 1.999, meaning the stated constant factor.

Output

First, you should output one number n, the maximal achievable number of loaves for the given value of the constant factor. Then, you should output any proof that this number of loaves is in fact achievable: n-1 descriptions of cutting, using the following notation. At each step, you print two numbers: first, the index of the loaf that you want to cut into two parts; second, the length of the newly created loaf (cut off from the original one). It is assumed that the starting loaf has index 0. Each newly created loaf will be given the lowest possible free integer index (so, at the ith step this will be i). Each time, the size of size of the original loaf will be decreased by the size of the newly created loaf.

Example

Input:

1.5

Output:

4

0 0.4

0 0.3

1 0.2"

## Your program will take as an input 'n' coordinates of type {(X1,Y1,Z1), (X2,Y2,Z2), (X3,Y3,Z3),...(Xn,Yn,Zn)} and from these 'n' coordinates print a list of 's' coordinates (where 's' is another input parameter less than 'n') which are closest to the origin (0,0,0) and a list of 't' coordinates (where 't' is another input parameter less than 'n') points closest to each other. Your solution should use an optimal strategy and minimal time / space complexity

## "int \* p= NULL; p = (int\*) malloc(0); what will be the value of p?"

## "if i write main like the following ways. Which one will compile and way?

1. char \* main();

2. char \* main(int, char);

3. char \* main(char);"

## "to print

\*

\*

\*\*

\*\*

\*\*\*

\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*"

## How will u find the min element in the stack with O(1) . Space constraint also has to be considered while designing it .

## Give an string, return the first non-repetitive character.

## "Consider an array of positive and negative integers. We want to find a slice of this array (i.e. a sub-array of consecutive elements) with at least two elements, such that the sum of the elements in this slice is equal to 0. The size of the slice can be anything (i.e. from 2 up to the length of the original array), and we don't care about finding the first, last, shortest, or longest slice, we just want a slice. Example: from [2,3,-1,2,-4] we would like to find the slice [3,-1,2,-4], where 3 + (-1) + 2 (-4) = 0"

## Find all the prime factors of a number entered

## Find the largest prime factor of a number

## You have a scooter which needs two tires. You are given three tires. Each tire has a max life of one year. What is the max time you can run your scooter?

## If in a tree , a node can have more than 2 children , is it possible to traverse it in inorder ? If yes , then how would it be done ?

## "What are the three types of basic variables in Perl?

## What would you use to see if a file exists?"

## What is "group by"?

## Difference between inner join & outer join

## Difference between final & finally

## What is the significance of virtual destructor?

## Discussion on virtual inheritance

## "Different ways to pass parameters to a function (by value, by reference, by pointer). for the following cases.

6.1 Basic data type (int, char etc)

6.2 Array of integers

6.3 an object of Structure

6.4 an object of a class

Discussed about possiblities of passing constant argument values (by reference, by pointer etc) & their syntax"

## What is the effect by keeping a constructor private? (in terms of inheritance)

## Diference between deep copy & shallow copy

## What are local static variables? Its usage & comparison with global static variables & class level static data members

## Difference between Malloc vs Realloc

## Given a regular expression and a text, find the min no of replacements to be done.

## Check if there is a common node in 2 linked lists.

## How to find min element in a stack in O(1), where the only operations are Push and Pop.

## Swap two numbers without using a temporarily variable.

## Design an algorithm and write code to find the common ancestor of two nodes in a tree

## Describe what a singleton is and when you would use it.

## write a program that accepts two mandatory arguments without using any built in date or time functions . The first argument is a string "[HH:MM {AM|PM}" and the second argument is an integer which denotes minutes. The minutes get added to the string. The return value or output of the program should be a string of the same format as the first argument. For example AddMinutes("10:23 AM", 13) would return "10:36 AM

## How do you programitcally create & terminate a process on linux

## What's the difference between mergesort and quicksort? When would you use each?

## Implement the stack in such a way that push,pop and minimum find in the stack operations are o(1).

## You have a large text file. Given any two words, find the shortest distance(in terms of number of words) between the two given words in the file. Can you make the searching operation in O(1) time? what about the space complexity for your solution?Ex:File contains:as was is the as the yahoo you me was the andWords given: the, wasAnswer should be: 0If Words given: you, theAnswer: 2

## "Recently in one of my interview i faced a question with Cube.The question goes like this. Draw a cube,on each of the 8 nodes binary numbers is represented.(am helpless since, actually diagramatic presentation will make it sense) The interviewer now wants to know why or in what fashion he represents the nodes with the binary digits?"

## Find the Mth last element of a singly linked list in the best possible time and most efficient use of memory.

## Given a Binary Search Tree, find its depth.

## You have a billion urls, where each has a huge page. How to you detect the duplicate documents?

## Given a value and a binary search tree.Print all the paths(if there exists more than one) which sum up to that value. It can be any path in the tree. It doesn't have to be from the root.

## Design a chat server

## Describe TCP Hand-shake and optimize SQL.

## What is a recursive mutex lock?

## What's hashmap in STL? How do you use it? When would you use it?

## What's the difference between a pointer and a reference?

## Explain the following terms: virtual function, virtual class, pure virtual function

## How do you distributed hash tables work?

## Difference between a mutex and a semaphore. When can we use one but not the other?

## How would you sort an array that consists of only zeros and ones in only one pass?

## How do static variables differ in C and C++?

## How would you bring the effect in c++ of static functions without using static?

## What data structure did you use on your previous project? Why? How would you improve on that?

## How would you bring the effect in C++ of static functions without using static

## "Given a dictionary of millions of words, write a program to find the largest possible rectangle of letters such that every row forms a word (reading left to right) and every column forms a word (reading top to bottom).

Anyone got any ideas?"

## Design a hash table to store phone #s. Your job is to write a hash function that has a parameter username, and generate a key. Username is unique, length 5 and can be A-Z, 0-9, space. Write a hash function that generate keys without collisions and use minimum memory.

## Coding: Reverse linked list

## How would the Destructor for Singleton look like ?

## Use of virtual destructor

## Practical usage of STL set.

# Amazon 1- 10 Questions

## Given a 2d matrix with characters and a dictionary. Find all the valid words in the 2d matrix. The words can be towards right, left , up or down. Exact code to be given.

## Given a sorted array, write a function to search first occurrence of a number in the array. If not found return -1.

Example::

{2,2,2,3,4,5,5,6,6,8,9}

search 6

should return 7.

## For a given BST, connect each of its right child to its inorder successor.

## There is a binary tree of size N. All nodes are numbered between 1-N(inclusive). There is a N\*N integer matrix Arr[N][N], all elements are initialized to zero. So for all the nodes A and B, put Arr[A][B] = 1 if A is an ancestor of B (NOT just the immediate ancestor).

## Given a file of n lines, where n-1 lines are identical and 1 line is different. Find the unique line in not more than one scan of the file.

## There are exactly N advertising boards on the highway. Now a company want to advertise on some of these advertising boards (each advertising board costs some money).

Company strategy is that, they want at least 'K' advertisement should be there among M consecutive advertising boards. But at the same time Company want to pay minimum for its advertisement.

Now, what is the total number of ways Company can advertise meeting its minimum cost strategy.

Also 1 <= K <= M <= 50 and M <= N <= 10^9

As for Example: N = 3, M = 2, K = 1 ==> there is only one way for minimum cost, ie. 0C0 , where '0' denotes No company advertisement, and 'C' denotes company advertisement board.

Similarly, for N = 4, M = 2, K = 1 ==> there are 3 possible ways, ie. C0C0, 0C0C, 0CC0.

## How would you test each of the 3 layers in a 3-tier web app?

## How would you test amazon search functionality on the home page?

## Given an array with different numbers and a number of C,so how to find all the combinations which the sum is C..like..array={1,2,3,4},C=3,,return is 2,which contains two combinations{{1,2},{3}}.

## You visit a website and it is slow today (it is not slow everyday). What could be the cause(s) of the slowness?

## You have an array of binary numbers as ""00001101000001010100000...""... We need to find the First occurrence of 1 in this series.. using binary search.

we need to design an algorithm of complexity less than O(n).. and we need to use binary search strictly..

## Design a chess game. Write all classes and methods.

## Given a large file of (x,y) coordinates. Find the k farthest points from origin.

## Given a network of printers and systems. Allocate the nearest printer to each system. How will you handle dynamic addition of printers and systems.

## Given a sorted array which is rotated n number of times. Find out how many times the array is rotated. Time complexity should be less than O(n).

## Given a Binary Search tree along with the parent pointer, find the next largest node for the given node. Give the time and space complexity. The node Structure is

class Node {

Int data;

Node left;

Node right;

Node parent;

}

## Given a string in the form of a Linked List, check whether the string is palindrome or not. Don’t use extra memory. Give the time complexity. The node structure is

Class Node {

Char data;

Node next;

}

## given an array of charactes have to replace space with %20. where %20 is considered as 3 characters.write complete code to implement this.

ps: assume that array has enough space at the end that can fit one space character to 3 chacters(%20).

## Find the maxProduct of three numbers from a given integer array.

## For 2 given array a[] and B[], find the highest index of A such that logical array A[0...i] and A[N-1...N-1-i] are same.

## Given an array of integers such that each element is either +1 or -1 to its preceding element. Find 1st occurrence of a given number in that array without using linear search.

## Function to reverse a c style sub-string

start - points to the first character to be reversed

end - points to character after the last character to be reversed

Note: STL not allowed

void reverse(char\* start, char\* end)

## 1. N-Petrol bunk problem: There are n petrol bunks located in a circle. We have a truck which runs 1 km per 1 liter (mileage 1kmpl). Two arrays are given. The distances between petrol bunks are given in one array. Other array contains the no of liters available at each petrol bunk. We have to find the starting point such that if we start at that point , you we would able to visit entire circle without running out of fuel. Initially truck has no fuel

## write a code to print the second largest element in a list

Shortest possible complexity.

## There is a HealthMonitor and two Servers (Primary and Secondary), all connected to one and another.

The HealthMonitor keeps pinging both the servers at specific time intervals and waits for their response for a time-out period after the request has been sent.

The server responds with a health status of itself and of its neighbor (meaning Primary responsds: OK; NEIGHBOR\_OK)

Implement the server's code to send and receive responses and then take action based on response.

## Write a class that will have following functions:

long CheckOut()

CheckIn(long)

Range of values is 1 to LONG\_MAX

At any given point in time checkout should return the minimum available LONG number

Checkin can return the value back

No need to check for border conditions (e.g. check out when all values are exhausted)

Implement:

1. long checkout()

2. void checkIn(long input)

## Write a class For Contacts on a device

Implementing Search a contact was the biggest problem I faced (because search should potentially search: FirstName, LastName, Address, PH#, Email etc)

## Write a class for a parking garage:

One level

One entry point

No membership or payments required

Handles multiple types of cars

## Implement:1. a search that will return all the strings that match a sub-string 2. an insert into this datastructure

Class

{

Insert (string str){};

List<strings> Predictions(string subString){};

}

## Implement an iterator for a Binary tree. It should have the following things:

1. bool HasNext()

2. <T> Next()

It should be an in-order traversal.

## If a function is given mostCommonChar(String str, int num) ,

1-.First input is Aabra Ka Daabra and second argument is 1 then the function should return first most repeated character in the string .Means in sorted descending .

2-> First input is Aabra Ka Daabra and second argument is 2

then the function should return second most repeated character in the string

like wise 3rd 4rth ....etc

## Write a C program to find the number of shift required to convert one string to another. Check all the corner cases.

Eg: abc to acb o/p shd be 2 as 'b' shifted from 1st index to 2nd and 'c' shifted to 1st from second.

## Adding Very Large Numbers. Write clean code for it. please check all corner cases..

Number can be really really large

## Design a singleton design pattern.

## Design a class in C++ such that only one object of it can be created.

## Find the 3rd closest element in a bst.You will be given a pointer to root and a value within the tree against which the closest has to be figured out. (closeness is in terms of value, not by distance ) and then follow up qn: for finding the kth closest in a bst.

## A video streaming server is generating the following data. Find the potential customers facing buffering issues.

A person is said to face buffering issues when he hits the play button multiple times on the same video

You are given a huge file (say 1GB) that contains the following data:

CustomerId-TimeStamp-Event-VideoId-Videolength

0040 -01.00pm -Play -Video1 -02:30:00

Write code for this. What data structure will you use

He also said, lets say all the parsing is taken care of and you are given a collection of classes that contain the above data:

Class

{

CustomerId

TimeStamp

Event

VideoId

}

## Design an online hotel reservation system.

## Design a furniture store with Tables and chairs. Write a constructor for chair and table

## You are given a UNIX path with dot (current) and two dots (parent). Convert this to an absolute path

E.g. $/home/abc/.././def/./ghi/../.

becomes $/home/ghi/

## Input is given a binarytree and out is sum of the all the children data and its node data .

Eg:

1 28

2 3 11 16

4 5 6 7 4 5 6 7

## Given a sorted array consisting 0's and 1's. find the index of first '1'. write a complete program which takes less time complexity. and test all boundary conditions also.

Eg: If given array is 0,0,0,0,0,0,1,1,1,1 the out put should be 6.

## Explain Collaborative Caching?

## There are lots of string in a file. Find the longest string that could be made from the other strings in the file.

Eg.

the

there

after

thereafter

reaf

ans.

thereafter

## Given an array A of length n where each element is 1..k where k is much smaller than n,

find one set of distinct indices i1 and i2 such that A[i1] + A[i2] == z.

## Code to create a file system.... Have classes like directory, file and all

please write the full code

## In a BST, I want to replace all nodes with value which is the sum of all the nodes which are greater than equal to the current node.

5

2 10

Output -->

15

17 10

## You are given a BST, and min, max elements. Your task is to trim this BST so that it contains the elements between the min and the max elements.

For example, given the mix and max elements [5, 13] and the tree below, you would return the output below.

8

3 10

1 6 14

4 7 13

output should be :--->

8

6 10

7 13

## Design a DS for storing browsing history.

## Design a Calculator. Details about the class variables, datastructure to be used etc.

## Generate a solution when multiple threads want to just read in their critical sections and when nobody is writing in the critical section.

## Finding border of a binary tree.Given a Binary tree print all the nodes that form the boundary.

## For given N\* N matrix,

1 2 3

8 9 4

7 6 5

Write a program to

print 1,2,3,4,5,6,7,8,9

## how do you handle your thread, to avoid dead lock and efficient(generally question)

## what is Materialized view, is any different from View.

## Write a program to swap kth node from first and kth node from last in a linked list .

## Design a MMORPG game in internet scale. Assume only available action for the players is watch and move.

## Given a matrix that contains 0s and 1s, find the shortest exit and print the path. You can navigate in top,bottom,left or right directions.

## In a stream of numbers, keep track of 1 million max numbers.

## find the Langford sequence for a given N if it exists? Details of Langford sequence - https://en.wikipedia.org/wiki/Langford\_pairing

## Given an array of integers,write a function that retrieves unique instances of any duplicates, returning them in a new array -

[2,1,2,4,3,1,5,1]

= [2,1]

[1,1,1,1,1,1,1,1,1]

=[1]

Write test cases for this function

## In a shop, product X is available in different quantities q1,q2,q3...... with price tags p1,p2,p3,...

wap to purchase X of quantity Q such that total price is less and also number of baggage is less

\*Consider the cost to be optimum than baggage.

(Sorry the trouble guys, I have edited the ques here)

## Wap to find kth largest element in a binary search tree

## Given a binary tree convert it to doubly linked list, with left pointer of binary tree as prev pointer of doubly linked list and right pointer of binary tree as next pointer of doubly linked list.

Example:

Input:

Binary tree with

A as root

B left of A

C right of A

B&C have no children

Output:

B->A->C->null

null<-B<-A<-C

## Given a linked list, print n nodes from tail of the list in reverse order

Example:

1->2->3->4->5->6->7->8->9->10

Output:

n=3

10->9->8

## How can you implement a Hashtable with any given data structure

What is hash function

How can you resolve collisions

## Given an integer array, return the combinations of 4 array values whose sum is x

Eg:

Input int array = {1,2,3,5,0,-2}

Return all possible combinations such that

a+b+c+d = 1

Like: -2 , 0 , -2 , 5

2 , -2 , 0 , 1, etc...

## Create a Session Manager class that iterates thru Session objects throw stale sessions out - How would you automatically purge Session objects that have not been active for 30 seconds or (n) seconds? The answer needs to handle millions of sessions.

## Youtube is not playing the selected video. What could be the problem and how to debug this issue?

## Write a program to check whether a substring is present in a main string.

## Write all possible test cases for SMS (Short Messaging Service) on a mobile device.

## Data Structure for node of linked list

Find the nth node form end of a linked list and also write test case

i have given solution with o(n) after that they asked to me how to break the my algorithm

## Design an algorithm to remove the duplicate characters in a string without using any additional buffer. NOTE: One or two additional variables are fine. An extra copy of the array is not. also do with o(n2),o(n),o(1) ,write test case and breake the algorithm whatever you write.

## Length is given as input.Print all possible permutations of numbers between 0-9.

Eg: if input length=4

all possible combinations can be 0123, 1234, 5678,9864,...etc all combinations of length from in all numbers between 0-9

## Create a basic implementation of Deferred.

Deferred#resolve => Marks the deferred / promise as completed successfully and calls success callbacks

Deferred#reject => Marks the deferred / promise as failed and calls failed callbacks

Deferred#promise => Returns promise object for watching completion

#addCallback => Adds a callback function to be called if the activity is successful

#addFailCallback => Adds a callback function to be called if the activity fails

var promise = myAsyncThingy(); // Call the API that starts some async work and returns a promise

promise.addCallback(function () { console.log(""We did it""); }); // Register a success callback

promise.addFailCallback(function () { console.error(""We failed""); }); // Register a failure callback

function myAsyncThingy() {

var def = new Deferred(); // Create a deferred to use for managing our async behavior

setTimeout(function () { def.resolve() }, 1000); // In one second call the resolve function to mark it as success

return def.promise(); // return the promise for our deferred to the client

}

## Using the mythical Hydra as an example, create a button that is destroyed by clicking it, but two new buttons are created in it's place.

## Create a function that will reverse the words in a sentence.

## You have unique ASCII characters. How you can sort them ?

Run time complexity of approach ?

## N boys are sitting in a circle. Each of them have some apples in their hand. You find that the total number of the apples can be divided by N.

So you want to divide the apples equally among all the boys.

But they are so lazy that each one of them only wants to give one apple to one of the neighbors at one step.

Calculate the minimal number of steps to make each boy have the same number of apples.

Input Given:

1. A number N => number of children.

2. Sequence of N numbers, each representing number of apples a child has.

<<P.S.>>

Passing an apple means a child giving away one apple to one of its neighbour.

Even if 2 separate children can pass apples simultaneously or one child can pass 1-1 apple to each of its neighbours then that will still be counted as 2 steps and not 1 step.

## You are given with space of binary codewords, and you have to come up with an algo to generate all subspace of size 2^1 , 2^2 ,2^3 . . .2^n of that set.

subspace is defined as:

1. it should have the 00..000 code word.

2. it should satisfy closure property. ie if we add any 2 codewords then result shud lie in the subspace.

Note: code words are to be added simply under modulo 2. no concept of carry is there.

ie. 1111 + 1010 = 0101

## given an array A[0-n], find the combination of A[i] and A[j] such that

1) A[j] - A[i] is max

2) A[j] > A[i]

and

3) j > i

please suggest a better solution than n-square.

## Reverse words in a sentence.

Ex:

Input: ""reverse the word""

Output: ""word the reverse""

## How can you implement queue?

I told him about list, array, stack implementation of queue.

## Write an algorithm that will take two dates and tell you if they are more than a month apart, less than a month apart or exactly a month apart.

## Some theory

1. What is difference between override and overload

2. abstract. when will u use abstract

3. what is an interface

4. what is difference betwwen array and link list

5. what is a tree

6. what is a map\dictionary

7. Explain (orally) how would you implement a dictionary via a tree

## There are different buildings in one environment, each with machines that can handle one request at a time. How would you design the request handling so that there is no single fail-point and is scalable.

Hint: It is ok if a request is sent to a machine that is already servicing another request. We can handle requests that come back from a machine. But he didnot want a lock on a single file that contained the data of empty machines

Follow up question was, lets say BLDG-A has 250 free machines, BLDG-B has 500 free machines, BLDG-C has 100 free machines and BLDG-D has 0. How would you assign requests? What if you had 850 requests at the same time? Why would you assign what you did?

## Given a set of array of size n, return all possible subset of size k.

example: if arr = { 1,2,3,4,5,6} , k=2;

return result is: {1,2};{1,3};{1,4};{2,3};{2,4};{3,4}

or a single array {1,2,1,3,1,4,2,3,2,4,3,4}

## Given a sequence of non-negative integers find a subsequence of length 3 having maximum product with the numbers of the subsequence being in ascending order.

Example:

Input: 6 7 8 1 2 3 9 10

Ouput: 8 9 10

## Special Property Numbers:

Eg--> You have a number 8987656 or 4565676

The difference between consecutive numbers is either 1 or -1.

You are given a range, you need to print the numbers with this special property.

## This is on Additive Number Property

Additive Number examples:

123459 (1+2=3, 4+5=9)

314538 (3+1=4, 5+3=8)

122436 (12+24=36)

You are given a range, you need to print all the additive numbers.

## You are given a 2-D array with same number of rows and columns. You have to determine the longest snake in the array. The property to find the snake is the difference between the adjacent(left, right, up or down) should be either 1 or -1. If there are more than one snakes with maximum length, the output should print both of them.

Example-->

The given array elements are as follows:

4 7 9 8

5 6 5 4

6 7 8 5

10 9 7 6

The longest snakes are 7->6->5->4->5->6 and 7->6->7->6->5->4

## Class and Data Structure Design for a ""metric"" system to determine the top song of a band. Two Web Service calls:

void play(String bandname, String songname);

String topSong(String bandname);

CONSTRAINTS: For this exercise we should constrain the design to a single server and do NOT use a database, but in memory data structure.

SAMPLE INPUT/OUTPUT

play(""Guns N Roses"", ""Welcome To the Jungle"");

topSong(""Guns N Roses"") => ""Welcome To the Jungle""

play(""Guns N Roses"", ""Sweet Child of Mine"");

topSong(""Guns N Roses"") => ""Welcome To the Jungle""

play(""Guns N Roses"", ""Sweet Child of Mine"");

topSong(""Guns N Roses"") => ""Sweet Child of Mine""

scale the architecture

## Design a Text Editor, in term of class diagram and data structure required to store the data/text and Insert, Delete, search in both direction and Edit operation. Calculate the time complexity for all operation.

Note: the size of file is huge and we don't have the RAM to load entire file into a memory.

## Design a Cache System, with the appropriate data structure and operation with the time complexity.

## Given an array find the next greatest element to the right of it. [4, 5, 2, 25]

Element NextGreatestElement

4 --> 5

5 --> 25

2 --> 25

25 --> -1

i gave o(n^2), but was

asked to solve in o(n)..You can take extra space...

## I was given a space of binary codewords containing 2^k codewords of word length n. And I was asked to generate all possible subspace of size 2^1, 2^2, 2^3 . . . . .,2^k.

Definition of subspace: It should have zero codeword, and it should satisfy additive closure property under modulo 2.

example: say given space of codeword is {0000,0001,1110,1111}. Then subspace of size 2^1 are D1{0000,0001} , D2{0000,1110}, D3{0000,1111}. and subspace of size 2^2 is {0000,0001,1110,1111}.

note: the additive closure is checked under modulo 2. ie. example1 1100 + 1010 = 0110 example2 111100 + 111111 = 000011

(no concert of carry was there, bits are added and modulo 2 is taken)

## Given an array of integers, find Pythagorean triplets.

i.e. find a,b and c which satisfies a^2 + b^2 = c^2

Integers could be positive or negative.

## Given a Directed graph, and a source point say S, and say C is the most distant node from S, you need to find the route from S to C . Suppose ,for example there is a path from A-> B- >C, and a path from A->B->D->C, here the most distant node is C with path length as 4. Also, each edge has unit length. Since the graph is directed , you need to implement an algo which takes care of the cycles in the graph. How would you implement this?

## Given a string, you need to find super string by word match. i.e. all words in the input string has to occure in any order in output string.

e.g. given data set:

""string search""

""java string search""

""manual c++ string search equals""

""java search code""

""c++ java code search""

...

input: ""java search""

output:

1) ""java string search""

2) ""java search code""

3) ""c++ java code search""

input: ""c++ search""

output:

1) ""manual c++ string search equals""

2) ""c++ java code search""

There are millions of records in given data set and you need to process few million as input.

## there is a log file which contains info in below format:

timestamp : customer-id : page-id

repeat customers are customers who return to the amazon site(any page) after at least a day.

write a code to print all the repeat customers

## Consider a city (visualize a circle). It has n petrol stations in it. You are given the maximum amount of petrol that can be filled at each of these stations. You are also given the distance between one station to the next one. The aim is to cover the entire city and come back to the start point. Assume that 1 liter of petrol will last for 1km.

List out all the possible petrol stations from where the journey can be started, so as to cover the city.

## You are given an integer array, where all numbers except for TWO numbers appear even number of times. Find out the two numbers which appear odd number of times.

## int board[8][8] each value in the matrix represents a character. 1-9 number represents all whites and 11-19 represents all blacks.

Given a pawn at (x,y) print all possible moves. Assume whites are index 0 and blacks are at index 7.

## The cost of a stock on each day is given in an array, find the max profit that you can make by buying and selling in those days

## A = {5, 3, 8, 9, 16} After one iteration A = {3-5,8-3,9-8,16-9}={-2,5,1,7} After second iteration A = {5-(-2),1-5,7-1} sum =7+(-4)+6=9 Given an array, return sum after n iterations

## You have a dictionary, D, that stores the positions of words in a document by mapping words (strings) to positions in the document (arrays of ints.)

You also have a list of words, L.

Your job is to find the shortest sequence of words in the document that contains all the words in L.

E.g., if the document is ""a b a c d x b a"", then

D[""a""] = [0 2 7]

D[""b""] = [1 6]

...

If we are given that L=[""a"", ""c"", ""x""]

Then we should return the start and end point of the shortest sequence that contains all words in L, which is (2, 5)

...the simple way is O(n^2) where n is the number of words in the document

...the way I came up with is exponential in |L|

...the interviewer had a way that was O(n)

## Write a function that finds out if any two numbers within that array add up to a target.

bool addsUp(Array<int> input, int target);

## Given a double ended queue with front and rear as two pointers, if a two people play a game. Say removal of number from a given list- from either side of the list. given the fact that the players can view the list of numbers, maximize the chance of each player winning?

## There are n coins in a line. (Assume n is even). Two players take turns to take a coin from one of the ends of the line until there are no more coins left. The player with the larger amount of money wins.

1. Would you rather go first or second? Does it matter?
2. Assume that you go first, describe an algorithm to compute the maximum amount of money you can win.

**Hints:**  
If you go first, is there a strategy you can follow which prevents you from losing? Try to consider how it matters when the number of coins are odd vs. even.

**Solution for (1):**  
Going first will guarantee that you will not lose. By following the strategy below, you will always win the game (or get a possible tie).

* Count the sum of all coins that are odd-numbered. (Call this **X**)
* Count the sum of all coins that are even-numbered. (Call this **Y**)
* If **X** > **Y**, take the left-most coin first. Choose all odd-numbered coins in subsequent moves.
* If **X** < **Y**, take the right-most coin first. Choose all even-numbered coins in subsequent moves.
* If **X** == **Y**, you will guarantee to get a tie if you stick with taking only even-numbered/odd-numbered coins.

You might be wondering how you can always choose odd-numbered/even-numbered coins. Let me illustrate this using an example where you have 10 coins:

If you take the coin numbered 1 (the left-most coin), your opponent can only have the choice of taking coin numbered 2 or 10 (which are both even-numbered coins). On the other hand, if you choose to take the coin numbered 10 (the right-most coin), your opponent can only take coin numbered 1 or 9 (which are odd-numbered coins).

Notice that the total number of coins change from even to odd and vice-versa when player takes turn each time. Therefore, by going first and depending on the coin you choose, you are essentially forcing your opponent to take either only even-numbered or odd-numbered coins.

Now that you have found a non-losing strategy, could you compute the maximum amount of money you can win?

**Hints:**  
One misconception is to think that the above non-losing strategy would generate the maximum amount of money as well. This is probably incorrect. Could you find a counter example? (You might need at least 6 coins to find a counter example).

Assume that you are finding the maximum amount of money in a certain range (ie, from coins numbered i to j, inclusive). Could you express it as a recursive formula? Find ways to make it as efficient as possible.

**Solution for (2):**  
Although the simple strategy illustrated in **Solution (1)** guarantees you not to lose, it does not guarantee that it is optimal in any way.

Here, we use a good counter example to better see why this is so. Assume the coins are laid out as below:

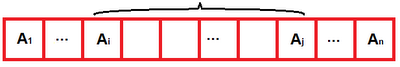
**{ 3, 2, 2, 3, 1, 2 }**

Following our previous non-losing strategy, we would count the sum of odd-numbered coins, **X** = 3 + 2 + 1 = **6**, and the sum of even-numbered coins, **Y** = 2 + 3 + 2 = **7**. As **Y** > **X**, we would take the last coin first and end up winning with the total amount of **7** by taking only even-numbered coins.

However, let us try another way by taking the first coin (valued at 3, denote by **(3)**) instead. The opponent is left with two possible choices, the left coin **(2)** and the right coin **(2)**, both valued at 2. No matter which coin the opponent chose, you can always take the other coin **(2)** next and the configuration of the coins becomes: **{ 2, 3, 1 }**. Now, the coin in the middle **(3)** would be yours to keep for sure. Therefore, you win the game by a total amount of 3 + 2 + 3 = **8**, which proves that the previous non-losing strategy is not necessarily optimal.

To solve this problem in an optimal way, we need to find efficient means in enumerating all possibilities. This is when [Dynamic Programming](http://www.ihas1337code.com/search/label/dynamic%20programming) (DP) kicks in and become so powerful that you start to feel magical.

First, we would need some observations to establish a recurrence relation, which is essential as our first step in solving DP problems.

[](http://4.bp.blogspot.com/_UElib2WLeDE/TVJNJgjHawI/AAAAAAAACdI/phIGr-4o_JM/s1600/coins.png)

The remaining coins are { Ai … Aj } and it is your turn. Let P(i, j) denotes the maximum amount of money you can get. Should you choose Ai or Aj?

Assume that P(i, j) denotes the maximum amount of money you can win when the remaining coins are { Ai, …, Aj }, and it is your turn now. You have two choices, either take Ai or Aj. First, let us focus on the case where you take Ai, so that the remaining coins become { Ai+1 … Aj }. Since the opponent is as smart as you, he must choose the best way that yields the maximum for him, where the maximum amount he can get is denoted by P(i+1, j).

Therefore, if you choose Ai, the maximum amount you can get is:

P1 = Sum{Ai ... Aj} - P(i+1, j)

Similarly, if you choose Aj, the maximum amount you can get is:

P2 = Sum{Ai ... Aj} - P(i, j-1)

Therefore,

P(i, j) = max { P1, P2 }

= max { Sum{Ai ... Aj} - P(i+1, j),

Sum{Ai ... Aj} - P(i, j-1) }

In fact, we are able to simplify the above relation further to (Why?):

P(i, j) = Sum{Ai ... Aj} - min { P(i+1, j), P(i, j-1) }

Although the above recurrence relation is easy to understand, we need to compute the value of Sum{Ai … Aj} in each step, which is not very efficient. To avoid this problem, we can store values of Sum{Ai … Aj} in a table and avoid re-computations by computing in a certain order. Try to figure this out by yourself. (Hint: You would first compute P(1,1), P(2,2), … P(n, n) and work your way up).

**A Better Solution:**  
There is another solution which does not rely on computing and storing results of Sum{Ai … Aj}, therefore is more efficient in terms of time and space. Let us rewind back to the case where you take Ai, and the remaining coins become { Ai+1 … Aj }.

[](http://4.bp.blogspot.com/_UElib2WLeDE/TVJUaw3myeI/AAAAAAAACdM/-zYRDclcIlI/s1600/coins2.png)

You took Ai from the coins { Ai … Aj }. The opponent will choose either Ai+1 or Aj. Which one would he choose?

Let us look one extra step ahead this time by considering the two coins the opponent will possibly take, Ai+1 and Aj. If the opponent takes Ai+1, the remaining coins are { Ai+2 … Aj }, which our maximum is denoted by P(i+2, j). On the other hand, if the opponent takes Aj, our maximum is P(i+1, j-1). Since the opponent is as smart as you, he would have chosen the choice that yields the minimum amount to you.

Therefore, the maximum amount you can get when you choose Ai is:

P1 = Ai + min { P(i+2, j), P(i+1, j-1) }

Similarly, the maximum amount you can get when you choose Aj is:

P2 = Aj + min { P(i+1, j-1), P(i, j-2) }

Therefore,

P(i, j) = max { P1, P2 }

= max { Ai + min { P(i+2, j), P(i+1, j-1) },

Aj + min { P(i+1, j-1), P(i, j-2) } }

Although the above recurrence relation could be implemented in few lines of code, its complexity is exponential. The reason is that each recursive call branches into a total of four separate recursive calls, and it could be n levels deep from the very first call). Memoization provides an efficient way by avoiding re-computations using intermediate results stored in a table. Below is the code which runs in O(n2) time and takes O(n2) space.

const int MAX\_N = 100;

void printMoves(int P[][MAX\_N], int A[], int N) {

int sum1 = 0, sum2 = 0;

int m = 0, n = N-1;

bool myTurn = true;

while (m <= n) {

int P1 = P[m+1][n]; // If take A[m], opponent can get...

int P2 = P[m][n-1]; // If take A[n]

cout << (myTurn ? "I" : "You") << " take coin no. ";

if (P1 <= P2) {

cout << m+1 << " (" << A[m] << ")";

m++;

} else {

cout << n+1 << " (" << A[n] << ")";

n--;

}

cout << (myTurn ? ", " : ".\n");

myTurn = !myTurn;

}

cout << "\nThe total amount of money (maximum) I get is " << P[0][N-1] << ".\n";

}

int maxMoney(int A[], int N) {

int P[MAX\_N][MAX\_N] = {0};

int a, b, c;

for (int i = 0; i < N; i++) {

for (int m = 0, n = i; n < N; m++, n++) {

assert(m < N); assert(n < N);

a = ((m+2 <= N-1) ? P[m+2][n] : 0);

b = ((m+1 <= N-1 && n-1 >= 0) ? P[m+1][n-1] : 0);

c = ((n-2 >= 0) ? P[m][n-2] : 0);

P[m][n] = max(A[m] + min(a,b),

A[n] + min(b,c));

}

}

printMoves(P, A, N);

return P[0][N-1];

}

## Find the next greatest number that can be form by same digit of the given number. sample test case input 4765 output 5467

## waf to swap kth node from first with kth node form last

example: if k=3

link list input 1->2->3->4->5->6->7.

output linked list 1->2->5->4->3->6->7

## WAP to find the longest arithmetic sequence in given array. Return number of element in series.

simple test case..

sizeofarray=4

array element

3 4 5 8

output

3

sample test case

size of array =10

array element..

-1 1 3 7 11 15 19 20 21 22

output

5

waf for a linked listed ..

shift all digit to first then consonant then vowel ..

such that list contain only one time one digit , consonant , vowel ..

List \*(List \*head);

input :

2->a->5->a->2->b->o->n->5->n.

output..

2->5->b->n->a->o..

wap to take one rotate a square matrix anticlock wise by 90 degree and add a particuler number after rotation to each prime column.

function prototype should be..

void rotate(int a[][],int size, int keytobeadded);...

sample test case....

input

size=3

keytobeadded=5;

square matrix :

1 2 3

4 5 6

7 8 9

output should be

3 11 9

2 10 8

1 9 7...........

Write a program that outputs all possible strings formed by using the characters 'c', 'a', 'r', ' b', ' o', and 'n' exactly once. Also write complexity.

Write an efficient function that returns the n’th Fibonacci number (There are many ways to solve this problem. Please write the most efficient method possible). Each Fibonacci number is the sum of the last two. The first 10 are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.

long getNthFibonacci(long i) { ... }

Setup:

Assume primitive Facebook. FB has Members.

class Member {

String name;

String email;

List<Member> friends;

}

Question:

Code printSocialGraph(Member m). Direct friends of m are Level 1 friends. Friends of friends are level 2 friends.....and so on

Print level 1 friends first. Then print level 2 friends....and so on

void printfriendsByLevel(Member m){

//Your code here

}

Let's say we're developing a vector graphics application. It will allow the user to create lines, rectangles, circles, text, etc. and manipulate them independently - move them, resize them, etc. Design an object model for this application. (How would you model the representation of the document in an object oriented language? What classes would you define? What methods would you have? What would your API look like?)

write a function that print TRUE if (){}[] are balanced in expression .. otherwise return FALSE.

[{()}] currect.. priority of [ >}>)..........it should be preserved

....................................................................................

void check(char \*a)

{

struct s

{

char ch;

s\*next;

};

s \*st=NULL;

s\*node=NULL;

if(a=='\0')

{

cout<<"TRUE";

return ;

}

else

{

while(a[0]!='\0')

{

{

node=(s\*)malloc(sizeof(s));

char c=a[0];

switch(c)

{

case '[':

if(st==NULL)

{

node->ch=c;

node->next=NULL;

st=node;

a++;

}

else

{

if(st->ch=='('||st->ch=='{')

{

cout<<"FALSE";

return ;

}

else

{

node->ch=c;

node->next=st;

st=node;

a++;

}

}

break;

case '{':

if(st==NULL)

{

node->ch=c;

node->next=NULL;

st=node;

a++;

}

else

{

if(st->ch=='(')

{

cout<<"FALSE";

return ;

}

else

{

node->ch=c;

node->next=st;

st=node;

a++;

}

}

break;

case '(':

if(st==NULL)

{

node->ch=c;

node->next=NULL;

st=node;

a++;

}

else

{

node->ch=c;

node->next=st;

st=node;

a++;

}

break;

case ')':

if(st==NULL||st->ch=='{'||st->ch=='[')

{

cout<<"FALSE";

return ;

}

else

{

if(st->ch=='(')

{

s \*newnode=st;

st=st->next;

free(newnode);

a++;

}

}

break;

case '}':

if(st==NULL||st->ch=='('||st->ch=='[')

{

cout<<"FALSE";

return ;

}

else

{

if(st->ch=='{')

{

s \*newnode=st;

st=st->next;

free(newnode);

a++;

}

}

break;

case ']':

if(st==NULL||st->ch=='{'||st->ch=='(')

{

cout<<"FALSE";

return ;

}

else

{

if(st->ch=='[')

{

s \*newnode=st;

st=st->next;

free(newnode);

a++;

}

}

break;

default :

a++;

if(a[0]=='\0')

{

cout<<"TRUE";

return ;

}

break;

}

}

}

if(st==NULL&&a[0]=='\0')

{

cout<<"TRUE";

return ;

}

if(st!=NULL)

{

cout<<"\nFALSE";

return ;

}

}

}

write a funtion to shift all vowel in first and all consonant in last . Order of all vowel and consonant should should preserved .

1)Given binary tree find the min weighted node. ?

Min weighted node is one which has minimun sum,where sum = Rootnode.getdata + leftnode.data + rightnode.data.

implement the method to achieve the above,,,

how would u implement a function tat would generate x object 20% of the time , y object 10% of the time,z object 70% of the time?

Which data structure would u use to pass this information to the function from the main.?

What is deque ? - double ended queue

Which data structures would u use to implement deque ?

Complexities of each data structure?

Implement it using array such tat all the operations are O(1).

operations are - insert at beginning,insert at end, delete at beginning ,delete at end.

How would u achieve this.?

Write interface and code to do the above.?

why value between twin primes is divisible by 6?

eg of twin primes are [5,7], [11,13], [17,19].

given a pointer to a node of sorted singly circular linked list and value to be inserted in that list, insert the value in sorted way.

Design a data structure to store names and phone numbers in mobile phone. Implement methods to insert and to get phone number .

Some numbers on phone keypad can represent several characters(such as 2-'A','B','C'). Design an algorithm to return a list of all vocabularies which represented by several numbers. You can implement the dictionary yourself.

Sample input: 227

output: car, bar, cap

The interviewer said he could solve it by O(1).

Someone know the answer?

Sorry for my poor English.

Given 2 arrays with numbers, multiply the numbers with corresponding indexes and return the sum of all the products.

Twist :- When one array gets consumed then start with its first element again.

A : 1,2,3,4,5

B : 2,1

Output: 24 (1\*2 + 2\*1 + 3\*2 + 4\*1 + 5\*2)

Print N numbers of form 2^i.5^j in increasing order for all i >= 0 , j >= 0 ?

Example : - 1,2,4,5,8,10,16,20.....

Goal: Create a software application that takes a set of people-relationships as input and provides a user interface that allows an operator to determine how “connected” any two people are.

Relationships / Input:

John:Jane:Jill:Jeb:Jim

Jane:John:Jason:June

Jill:John:Jim

Jim:John:Jill:June

Jeb:June:John

June:Jim:Jane:Jeb

Jason:Jane

For example, John has a 2nd degree connection to June since June has a 1st degree connection to Jim who in turn has a 1st degree connection to John.

In a series of 1 to N, two numbers are missing. Find the missing numbers? Quickest way?

Coding:

Q1: How do you make it thread safe?

I said use “public void synchronized” Good. But terrible performance since the entire method is synchronized.

Q2: Can you not lock on the entire method? I said used nested locks:

His q: This will lead to a deadlock if in another thread I call Transfer (id2, id1) and Transfer (id1, id2).

How do you prevent this then? How do you design your code to not to get in to deadlock? (stumbled here)

How do you detect deadlocks? What tools would you use? I said do “Kill -3 .<process id>, and analyse if anything is deadlocked.

3) Coding question:

Will the thread spawned will ever see the flagRun=false?

My corrected answer after a couple of attempts: No, since each thread will get a copy of it’s own flagRun, changing the flagRun value in the main thread will not be seen in the spawned thread.

How to fix it: declare flagRun to be volatile so that the values can be changed and seen in either threads.

What is difference between "volatile" and "static volatile"? Give an example

Explain what is “static synchronized?” What does it lock on? what is ‘synchronized’? what does it lock on?

Coding: Write a Client/Server. Three methods are given. Msg.Get(), Msg.Process(), Msg.Send(). Write code. Since Msg.Get() and Msg.Send() has to send messages over the network. It takes a lot more number of threads. So how many threads out of 10, would allocate to each of the three processes. What is the proportion?

Puzzle: There are 5 slots(1,2,3,4,5) and 5 people(A,B,C,D,E). Each of them provide their preferences. A=1, B=2, C=3, D=4, C=4. Given an arbitary starting sequence say BCDEA, going clock wise: how many passes does it take to fill in those slots so that max number of people are happy. People are happy if A gets 1, B gets 2, C gets 3, etc. Remember D and E cannot be kept happy together at the same time because both of them prefer Slot-4.

My answer: Worst case scenario when you start at BCDEA(only E is happy because E comes in 4th position and prefers 4), next rotation pass CDEAB(no one is happy), DEABC(no one), EABCD(no one), ABCDE(4 people are happy). So for N people, it takes N passes.

His question: Can you do better than N passes? (Can you do this in less than N passes?)

(I tried to come up with something but stumbled.)

Code up a simple Bloom Filter for four letter words. Assume all words are lowercase, exactly four letters long, and only use the letters a to z. An example set of test words might be able, band, bend, card, darn, form, grab, harm, look, make, mate, peer, and team, but assume the actual list of words is much bigger.

Each time a visitor requests a page from our website, our webserver writes a log entry recoding the visitor's identity and the kind of page requested. Entries are written in chronological order to a plain-text file, with one entry per line. The format of each entry is:

user-id page-type-id

User IDs are arbitrary strings that uniquely represent a given user; if a user visits multiple pages, each log entry will have the same user ID. Page type IDs are arbitrary strings that uniquely represent a given kind of page on our site, such as the homepage, a product detail pages, or the shopping cart. Tons of users visit our website, but there are only a few dozen types of pages.

We can use our weblogs to answer questions about user behavior. One interesting question is: what is the most common three page sequence through the site? E.g., if the most common pattern is to buy items advertised on the home page of the site, we might see the most common three page sequence as "Homepage -> ProductDetailPage -> ShoppingCart". However, if customers spend a lot of time browsing the "Customers who bought this item also bought" feature, we might see the most common three page sequence as "ProductDetailPage -> ProductDetailPage -> ProductDetailPage".

Attached is a sample log file for your reference. Within the first 10 lines of the sample, customer "234" travels through the sequences "Listmania -> ProductDetail -> Checkout" and "ProductDetail -> Checkout -> HomePage" once each.

For the sake of this test feel free to assume that everything will fit in memory. Do keep in mind that given the size of our data sets, performance has to be considered, also, we will be looking at more than just correct output..

How do you add up a very long list of numbers multiple threads (100 threads?) How many cores do you require?

How do you increase the performance of this program? Does the number of threads created get limited by the cores of the box? How exactly are you gonna delegate it to the cores? (as in- these number of threads need to use core#1, and so)

There is an old dry well. Its sides are made of concrete rings. Each such ring is one meter high, but the rings can have different (internal) diameters. Nevertheless, all the rings are centered on one another. The well is N meters deep; that is, there are N concrete rings inside it.

You are about to drop M concrete disks into the well. Each disk is one meter thick, and different disks can have different diameters. Once each disk is dropped, it falls down until: \* it hits the bottom of the well; \* it hits a ring whose internal diameter is smaller then the disk's diameter; or \* it hits a previously dropped disk. (Note that if the internal diameter of a ring and the diameter of a disk are equal, then the disk can fall through the ring.)

The disks you are about to drop are ready and you know their diameters, as well as the diameters of all the rings in the well. The question arises: how many of the disks will fit into the well?

Write a function:

class Solution { int falling\_disks(int[] A,int[] B); }

that, given two zero-indexed arrays of integers - A, containing the internal diameters of the N rings (in top-down order), and B, containing the diameters of the M disks (in the order they are to be dropped) - returns the number of disks that will fit into the well.

For example, given the following two arrays:

A[0] = 5 B[0] = 2

A[1] = 6 B[1] = 3

A[2] = 4 B[2] = 5

A[3] = 3 B[3] = 2

A[4] = 6 B[4] = 4

A[5] = 2

A[6] = 3

the function should return 4, as all but the last of the disks will fit into the well. The figure shows the situation after dropping four disks.

Assume that:

N is an integer within the range [1..200,000];

M is an integer within the range [1..200,000];

each element of array A is an integer within the range [1..1,000,000,000];

each element of array B is an integer within the range [1..1,000,000,000].

Complexity:

expected worst-case time complexity is O(N);

expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Given a sorted array of size N and a sorted array of size M+N, merge the first array into the second preserving order. There is enough space in the second array to hold all elements from the first one.

Suggest a best data structure which cab be used to represent N-ary tree. Given the root node of the same, write a function which finds out whether or not the tree has a loop/cycle.

Given a binary tree. Write a function that takes only root node as arguement and returns (sum of values at odd height)-(sum of values at even height).

Given a string, find the longest sub sequence which contains only unique characters.

Convert a BT into SUM BT(each node values = sum of left and right node).

Given a BST, along with left and right pointer for a node, it has forward and backward pointers, convert the tree into Doubly linked list using these extra pointers.

An array contains two sub- sorted arrays. Give an inplace algorithm to sort two sub arrays.

for ex: I/P: 1 4 5 7 8 9 2 3 6 10 11

O/P: 1 2 3 4 5 6 7 8 9 10 11

Write a program to sort an array of strings so that all anagrams are next to each other

ex

input {god, dog, abc, cab, man}

output {abc, cab, dog, god, man}

What should be the output of the following code.

class Test {

public int i=0;

@Override

public int hashCode() {

return i;

}

}

Class a{

psvm(){

HashMap <Test, String> hm = new HashMap();

Test t1 = new Test();

hm.put(t1,”success”);

sysout(hm.get(t1)); //print success

t1.i = 10;

sysout(hm.get(t1)); //NULL

}

}

Given an 2D array of characters. Find words in the array (either vertical or horizontal). a character cannot be part of 2 words. Maximize the number of characters used. Hint: 1D variant can be solved by Dynamic programming in linear time.

Find kth Largest element in BST

Write a code to find subset of numbers in array whose summation= given number? If this does not exist, print false

I know the solution consisting of non-negative numbers.

Is there any smart solution when the array contains NEGATIVE numbers?? Thanks.

Design a maze. Give the algorithm and code to find the correct path and get out of the maze.

Give the algorithm and code to get the depth of the deepest odd level leaf node in a binary tree.

Rotate matrix in 90 degree clockwise.

given a string in form of an array find an expanded string

e.g. A2B3C4 => AABBBCCCC

also, size of given array is exactly same as expanded string. so return the same array with expanded string.

How to Reduce wait time in Elevator Design System?

I am just curious to know if a greedy approach will work fine here?

Gave a string of characters and asked them to store in a binary search tree in such a way that it can be extracted in exactly the same order

Assume I have a log file with list of people with their arrival and departure time at an event that happened in the past.

My task is to find out the maximum number of people present at any time during the entire event? I am not given query time.

ai = Arrival time of person i

di = Departure time of person i

I have a list of pairs like (a1,d1), (a2,d2), (a3,d3).... (an,dn)... It's not in a database.

I apologize as I cannot edit my previous question. I think it had a incomplete description.

Please let me know if you guys still need clarification. Thanks

I was asked to design a meeting scheduler, just like in the Microsoft outlook calendar or the gmail calendar. I proposed that I will create an array of 48 for each day. Every 30 min representing the array entry.

I have to make sure that the next appointment does not collide with a previous meeting.

My solution works fine but it wastes too much memory.

Can anyone please tell me how do I find a better solution to detect collision for meetings.

I don't know all the meetings at the beginning. They will be added randomly later.

Thanks,

Find the longest common subsequence of two string

Discuss the design of the following game:

The board consists of the cells of 3 kinds:

ant,

water,

food

if the ant moves to the cell that has ant then both ants are destroyed

if the ant moves to the cell that has water, ant disappears

if the ant moves to the cell that has food, food cell become ant cell.

there are two players, game server.

Write a method that multiplies two integers without using multiply operator

What are the common problems in multithreading programming? Lock, Dead Lock

Discuss IEnumarable/IEnumerator pattern. WAP that implements the pattern for the collection with elements that are collections as well, i.e.:

((1,3,4), (4,5,6), (7,8,9))

input:

sum - the integer amount

n - the number of coins

d - the array contains the coins denominations

WAP that prints all the possible non-repeated combinations of n coins of denominations from d that sum up to n.

Sample of input:

d={1,5,10,15}

sum = 30

n = 6

The expected output:

1,1,1,1,1,25

5,5,5,5,5,5

discuss restaurant reservation system design

Serialize in a file and deserialize a binary tree.

Given a BST, print the node with value between [min, max], i.e. min = 10, max = 17, print the node with value between 10 and 17

Find the common friend between two people in facebook? Use graphs.

how to find lowest common ancestor of a binary tree

??

not BST

Provide a datastructure that can perform :

1. insert

2. delete

3. find min

4,. find max

5. delete min

6. delete max

all in O(1) time.

Design a scheduling system which can run 10M jobs. Jobs can be scheduled to run on a time of day, some day of the week, a particular day of the month etc.

A robot is placed on an infinite 2D grid. The robot is initially facing the east direction. It moves in a spiral movement turning to its left after each move. The movements are given as an input array.

For example, assume the Robot is initially at (x,y) and the movement array is [4,3,5,2,1,6,...].

After 1st move, Robot will be at (x+4,y)

After 2nd move, Robot will be at (x+4,y+3)

After 3rd move, Robot will be at (x-1,y+3)

After 4th move, Robot will be at (x-1,y+1)

After 5th move, Robot will be at (x,y+1)

After 6th move, Robot will be at (x,y+7)

and so on.

Find the minimum area of the rectangle which can enclose all these points.

You have a bunch of files and folders, Design a playlist which can have any file from any folder and a player that plays it

Count the number of shapes in a given (1/0) matrix. A cluster of consecutive (not diagonal) 1's defines one shape.

eg

1 1 0 0 1

1 0 0 1 0

1 1 0 1 0

0 0 1 0 0

No of shapes = 4

Write a function which compress string AAACCCBBD to A3C3B2D

Given a list of quantity of books and corresponding prices.User wants to purchase Q quantity of books. Provide an algorithm which suggest user Q Quantity of books with minimum price

e.g

Quantity of Books 10 20 30 15 25

price of books 100 200 120 130 165

user wants to purchase 500 quantity of books.come up with minimum price

Given an array sort all the elements in even positions in ascending order and odd positions in descending order

Given an array elements, Find the maximum number which can be formed by the array elements

Eg input – a[ ] = {9,6,8,1]

Output - 9861

A tree-map is implemented using BST, the complexity of search in a tree-map is guaranteed to be O(logn). How is that case of search complexity O(n) [obtained when the BST is like a linked list from the root node, only in single side] in BST avoided in tree-map.

Which data structure is preferred for performing concurrency, serialization out of BST and Hash-Table.?

Compare the space complexity of BST and Hash-Table.

Given a BST and a node, write a function to find the next biggest element in the BST in preferred language.

Suppose we are detecting fraud cheques and we found the cheques with the following list of patterns are fraud:

111122234455

1234

22334455

11111111

234567

etc.

Now if you have a new cheque and wan to detect fraud in O(1) time what data structure you want to use?

Suppose a customer buys items for $10 in a shop and the cashier swipe her card at a POS charging $10. Assume that the card has $100 balance before swiping. POS sends the $10 transaction to a machine A in the Amazon cloud. A calls a service to update transaction and card balance, and then sends acknowledgement back to the POS. But the ack got lost in the middle and POS sends another $10 transaction request. How would you make sure that the balance is $90, not $80. And how would you distinguish multiple try with two legitimate $10 transaction back to back.

Hint: You can't use more than one transaction entry in Database and you don't have the rollback provision.

Given a BST and a node, write the BST data structure and a method in Java

Node findNode(Node n)

that will find the next node of n in the BST. For example, if the tree looks like:

7

/ \

5 11

/ \ /

4 6 9

/ \

2 15

Then,

findNode(2) = 4,

findNode(4) = 5,

findNode(5) = 6

findNode(6)=7

findNode(7)=9

findNode(9)=11

findNode(11)=15

Note that you are not given the root of the tree.

Hint: you may assume that you have parent pointer.

what's encapsulation

Difference between bit-wise operator and logical operator

Design a GPS system

How to detect cycles in a graph?

Don't need to write code, just your idea and complexity.

what is volatile in java

What is IOC container and how did that works, and explaining the pros and cons.

Write a function, that given a list of integers and an integer s, prints any 2 numbers in the list that sum to s.

1, 2, 3, 4, 5 sum = 6 could print:

1 + 5 = 6

2 + 4 = 6

4 + 2 = 6

5 + 1 = 6

Given a file which contains list of function with parameters and return types (Assume parameters and return type as primitives) they can be of any order

Example: File functions.txt contains below info

int e=f1(a,b,c,d)

int c=f2(a,b,d)

f3(a,c)

float d = f4()

question follows as below

We need to implement function

execute(functions.txt,a,b,d) where functions.txt is the file which contains just functions info and a,b, d are input parameters of functions defined in the file.

Now read the file and we need to determine the order of execution of function depending on the input parameter list

we can't execute f1 first because we have only parameters a , b and d

f2 can be executed because we have a, b and d parameters, which gives parameter c

now f3 can be executed because we have parameters a and c..

so we need to find the order and execute functions as quickly as possible (can use multi threading)

so efficient order is

f2 and f4 can run parallel and then f1 and f3 can be started only after completion of f2.

What are the data structures we use here to solve this problem

Consider execute function which takes always 1st parameter as filename and the rest of parameters as dynamic (similar to ... in java for dynamic parameter list)

find the distance between 2 values in a binary search tree. Node will have value, left node and right node

implement the function int Distance(Node root, int val1, int val2) without recursion

Note: function depth(root, val) returns the depth of sub tree. Interviewer is ok about the depth of tree functionality but he asked me if Distance(Node root, int val1, int val2) implementation is correct.

Complexity is O(logN)

Given a multidimensional array with only 0s and 1s, reverse the array! I guess what he meant was flip 0s to 1s and vice versa! The array could be of any dimension ex:4X4X4X4....

(I dint get this q's so asked for another!)

Given a char array color[]={'a','b','c','d','e','f'.......'z'}

and a random array arr[]= {'f','a','b','b','z','a','a','a'}

you need to sort them such that resultant array will be {'a','a','a','a','b','b','f','z'}.

Conditions:

1) You should use swap function.

2) Every element may repeat minimum 5 times and maximum 26 times.

3) 'a' can be swapped only 1 time, 'b' can be swapped max 2 times,'c' can be swapped 3 times ...... z can be swapped max 26 times.

4) You cannot make elements of given array to 0.

5) you should not write helper functions.

Input will be of 100 elements each.

Suppose that we have a sorted singly linked list with integer values. For example:

1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7

We want to change the pointers of this linked list so that it becomes:

7->1->6->2->5->3->4

I did not have enough time left to finish my code and I could not think of a good solution.

Write a method to compute the difference between two ranges. A range is defined by an integer low and an integer high. A - B (A “minus” B) is “everything in A that is not in B”

Given N arrays with sizeof N, and they are all sorted, if it does not allow you to use extra space, how will find their common datas efficiently or with less time complexity?

There are two strings, String 1 and String 2 remove the characters from string 2 which are present in string 1

Example: String 1: Amazon, String 2: And

O/p: mzo

Given a sentence and a function isValidDictionaryword(String s), which returns true if the word is present in dictionary. WAP to separate valid dictionary words in the sentence with a space.

I/P : thereisastoneontheroad.

O/P : there is a stone on the road.

You are given two array lists. One Array List contains information of latitudes and longitudes of all the amazon stores and another array list contains all the possible values of latitudes and longitudes. Find an optimal way to find out all the latitudes and longitudes which are nearest to one pair of amazon store.

ArrayList<latitude, longitude> AmazonStore;

ArrayList<latitude, longitude> World;

Following is an interview question asked by 'Amazon' to me. I still haven't come up with a optimized solution.

Problem Statement:

Given an unsorted array of integers n. Return 'true' if the addition of any integers from that array matches with the target value else return false.

Note:

1)'n' could be 1000 or 10,000.

2) Target value could be 'negative'

Test Condition:

i/p:- Array A[]= {-5,6,7,1,0,12,5,-6,100}

Target = 13

o/p:- TRUE

As, 6+7=13.

If we try to do it linearly or normally it will take O(2^n) time complexity.

So I am looking for any method or algorithm which will optimized this problem more.

A spreadsheet consists of a two-dimensional array of cells, labeled A1, A2, etc. Rows are

identified using letters, columns by numbers. Each cell contains either an integer (its value) or

an expression. Expressions contain integers, cell references, and the operators '+', '-', '\*', '/'

with the usual rules of evaluation – note that the input is RPN and should be evaluated in stack

order.

Write a program (in C, C++ or Java) to read a spreadsheet from ‘stdin’, evaluate the values of

all the cells, and write the output to ‘stdout’.

The spreadsheet input is defined as follows:

• Line 1: two integers, defining the width and height of the spreadsheet (n, m)

• n\*m lines each containing an expression which is the value of the corresponding cell

(cells enumerated in the order A1, A2, A<n>, B1, ...)

Your program must output its data in the same format, but each cell should be reduced to a

single floating-point value. For example, we would expect the following expect to produce the

indicated output:

Input Expected Output

3 2

A2

4 5 \*

A1

A1 B2 / 2 +

3

39 B1 B2 \* /

3 2

20.00000

20.00000

20.00000

8.66667

3.00000

1.50000

The above example input visually looks like:

| 1 | 2 | 3 |

--+-------------+-------+--------------+

A | A2 | 4 5 \* | A1 |

--+-------------+-------+--------------+

B | A1 B2 / 2 + | 3 | 39 B1 B2 \* / |

------------------------+--------------+

Implement the Huffman compression algorithm as shown in the example below:

1) Given a string AAAAAABBCCDDEEFFFFF, group them according to the number of occurrences: A => 6, B => 2, C => 2, D => 2, E => 1, F => 5

2) Concatenate them according to their number of occurrences (adding the number of occurrences while doing it)

3) Put the concatenate strings in a tree-like structure:

Write a program to fill a crossword puzzle, given an empty 2D array, the starting positions of the to-be-filled words in the array and the length of each word and a dictionary of words.

Also, come up a appropriate DS for the words in the dictionary that can be specifically advantageous to this problem.

Given a 2D array of chars and a raw list of valid words.

1) Find all the valid words from the array. From each element in the array, you can traverse up, down, right or left.

Eg,

valid words from the above 2D array -> god, goat, godbody, amour,....

2) Also, find a suitable DS to store the raw words list.

I used a recursive approach to solve the problem in exponential time. Can't think of any better approach.

Given a list of n gas station of form P(D,X) where D is the distance from this station to next station and X is the amount of petrol available at this station, identify the starting station from where you can complete journey to each station in order from 1.....N. You can only go in one direction i.e from P(i) to P(i+1)

EDIT: I forgot to mention that travelling distance K consumes K units of gas.

EDIT2: I proposed an O(n^2) solution, then interviewer asked me if I can do better.

Given a linked-list and 2 integers k & m. Reverse the linked-list till k elements and then traverse till m elements and repeat.

For a given binary tree, print level order nodes in reverse i.e Bottom to Top in different lines.

Find the longest tree walk in a tree. The question is to print the longest path in a tree. the path need not got through the root. The path should be between two leaves.

There are 50 Red and 50 white balls. and two back.

Put all these balls in two bag such that if we pick any of the bag, probability of getting red is maximum .....

Defining a tree as such that the parent node always contains the sum of children nodes.

Coul be something like this):

private static int p\_get\_nodes\_value\_sum(TreeNodeCollection v\_tree\_original)

{

//TreeNodeCollection a\_child = v\_tree\_original.ChildNodes;

//TODO:sum the child nodes

int a\_suma = 0;

foreach (TreeNode a\_child in v\_tree\_original)

{

if (a\_child.ChildNodes.Count > 0)

a\_suma += p\_get\_nodes\_value\_sum(a\_child.ChildNodes);

else

a\_suma += Convert.ToInt16(a\_child.Value);

}

return a\_suma;

}

public class TreeParentSum: TreeNode{

public TreeParentSum(TreeNode v\_tree)

{

int a\_sum = p\_get\_nodes\_value\_sum(v\_tree.ChildNodes);

TreeNode a\_node = new TreeNode("PARENTNODE", a\_sum.ToString());

a\_node.ChildNodes.Add(v\_tree);

}

}

Change the structure of a Tree node to hold a pointer for the next in-order element (sucessor).

Given a Binary Search Tree.. transform it on a LinkedList by setting the next pointer described above.

Maybe this is a duplicate question.

Implement a shared int pointer in C++ (SharedIntPtr).. the use would be like this:

There is also this reference count which tells how many references do you have.

And the question was: Implement SharedIntPtr, constructors and reset().

Print all combination of given length k possible with characters available in a given string "S" with repetition in new lines.

Example

S="abc"

k=2

output:

aa

ab

ac

ba

bb

bc

ca

cb

cc

Create a list of Vertical sum of a given binary tree.

We are given a matrix of MxN elements with each element either being 0 or 1.Find the shortest path between a given source cell to a destination cell.

An element value of 0 means we cannot create a path out of that cell

You are given two Strings lets say "S" which consist Upper Case albhabets and '?' character only and p. You are required to replace '?' with some alphabet so that resulting string have maximum number of "p" in it. You can replace '?' with any alphabet.

2. Replace '?' such that it should be lexicographically sorted.

Example

S="ABAAMA????MAZON????????"

p="AMAZON"

The final string S = "ABAAMAZONAMAZONAAAMAZON"

S="?????"

p="ABCD"

Final Result="AABCD"

Soln:- Proceed from the end of the String.

Given a UNORDERED tree and two node elements, we are suppose to find the common LEAST ancestor if present from the given tree.

Given sorted integer array and a given value we have to find two elements which will sum up to a given value. Test all corner cases with minimum complexity.

An array of size N is given. Array is sub divided into sub array of size K. Find maximum value of each sub array.

My ans-

While traversing the array keep on adding values to max heap of size K and keeping a virtual window of size K on array.

When element leaves the window then remove the leaving element from heap too and reheapify the heap. And max element of that window will be again on top in heap.

Any better approach?

Given a node of Binary Tree . find all node's at distance k from it .

You have a web server's log that records for each user the URL that he accessed.

Example format:

How to find the maximum common (sub)sequence of visited URLs from all users? Write code in java

Log entries are ordered based on timestamp.

Example log (omitting timestamp for clarity):

John URL1

John URL2

Jim URL2

Mary URL1

John URL4

etc

Update:

Example:

So the max common URL sequence is A,C,C (URL A followed by URL C, followed by URL C)

Find if the given two trees can be joined leaf to leaf?

program to find all prime numbers in the range of

the input start/end numbers.

start number: 1,000,000,000,000

end number: 1,000,000,000,000,000

suppose i m sitting on a server and several machines are pinging me with a word each and we are storing these words in a Queue.Now,the data stored in the queue is very large and we can process it once only.At the end of the day we are given 5 words and we have to find all the anagrams(for each of the 5 words) present in the queue.

Given a Binary Tree (not BST) with integer values . 1) Find path from root to any node with max sum. 2) As there can be many path's find all of them. 3) Print all such paths.

Do this in O(n) : n is number of node's in tree. he wanted an O(n) solution not O(n)+O(n) ie. u can't traverse tree twice .

Classic 2-sum problem.

Hint, using hash table.

Implement atoi().

Select a random node data from a very long linked list whose length is not known such that the probability of each node is equal.

Large shared HDD is synced with Cloud Server. HDD contains same files multiple times by different users. How does the cloud server manage duplicates. It can't store all the duplicates. Duplicates may not be differ by file name. Discuss algorithm.

there are two arrays named A and B , both of them with k size, they are sorted in acsending order. could you find k-th smallest combinations of ai, bj -->(ai+bj) . 0<=i,j <k.

for example: a = {1, 3, 6} b = {4, 5, 6} then we will get 1 + 4 = 5, 1 + 5 = 6, and 1 + 6 = 7,the result is 5,6,7. does it make you understood? and could anybody do it with less time and space complexity.

Hi guys, thanks for all your suggestions and idea, and finally I get my answer and here are my c++ codes, time complexity is O(k\*lgk), and space complexity is O(k):

#include<iostream>

using namespace std;

typedef struct node{

int row;

int col;

int data;

}Node, \*PNode;

void swap(PNode &a, PNode &b) {

PNode temp = a;

a = b;

b = temp;

}

void adjust\_min\_heap(PNode \*bin, int i, int k) {

int left = 2 \* i + 1;

int right = 2 \* i + 2;

int min\_index;

if(left < k && bin[left]->data < bin[i]->data) {

min\_index = left;

} else {

min\_index = i;

}

if(right < k && bin[right]->data < bin[min\_index]->data) {

min\_index = right;

}

if(min\_index != i) {

swap(bin[i], bin[min\_index]);

adjust\_min\_heap(bin, min\_index, k);

}

}

void build\_min\_heap(PNode \*bin, int k) {

for(int i = k / 2; i >= 0; i--) {

adjust\_min\_heap(bin, i, k);

}

}

int \*get\_k\_th\_minimum(int \*a, int \*b, int k) {

PNode \*bin = (PNode\*)malloc(sizeof(PNode) \* k);

int \*result = (int\*)malloc(sizeof(int) \* k);

memset(result, 0, sizeof(int) \* k);

int i;

int count = 0;

for(i = 0; i < k; i++) {

bin[i] = (Node\*)malloc(sizeof(Node));

bin[i]->row = i;

bin[i]->col = 0;

bin[i]->data = a[i] + b[0];

}

build\_min\_heap(bin, k);

while(count < k) {

result[count++] = bin[0]->data;

bin[0]->col += 1;

bin[0]->data = a[bin[0]->row] + b[bin[0]->col];

adjust\_min\_heap(bin, 0, k);

}

for(i = 0; i < k; i++) {

free(bin[i]);

}

free(bin);

return result;

}

void main() {

int a[] = {1, 2, 4};

int b[] = {5, 9, 11};

int k = 3;

int \*p = get\_k\_th\_minimum(a, b, k);

for(int i = 0; i < k; i++) {

cout << p[i] << " ";

}

free(p);

getchar();

}

Suppose there is a distance matrix which consists of n points and that gives the distance of say (a,b) = 6 , (a,c) = 5. If there are N points assume 10000, then it requires N \* N matrix to store the corressponding distances.How to store the matrix in such a fashion that it gives a fast retrieval and optimized storage.

3.Find the first unique character in a Stream. Please note that you are being provided a stream as a source for these characters.

The stream is guaranteed to eventually terminate (i.e. return false from a call to the hasNext() method), though it could be very long. You will access this stream through the provided interface methods.

A call to hasNext() will return whether the stream contains any more characters to process.

A call to getNext() will return the next character to be processed in the stream.

It is not possible to restart the stream.

If there is no unique character, then return the character '#'. # won't be any character in the character stream.

You just have to complete the function getUniqueCharacter() using the functions hasNext() and getNext() which are already defined.

Example:

Input:

aAbBABac

Output:

b

Input:

aBBa

Output:

#

2.Given an integer linked list of which both first half and second half are sorted independently. Write a function to merge the two parts to create one single sorted linked list in place [do not use any extra space].

Sample test case:

Input: List 1:1->2->3->4->5->1->2; Output: 1->1->2->2->3->4->5

Input 2: 1->5->7->9->11->2->4->6; Output 2: 1->2->4->5->6->7->9->11

C/C++/Java/C#

struct node

{

int val;

node \*next;

}

node\* sortList(node\* list1) {

}

Java

class Node

{

int val;

Node next;

}

Node sortList(Node list1) {

}

1.You are given a function calcDifference which takes in the root pointer of a

binary tree as it's input. Complete the function to return the sum of values of nodes at

odd height - sum of values of node at even height. Consider the root node is at height 1

/\* Write your own custom functions here \*/

int calcDiff(node \* root){

/\*

struct node {

struct node \*left,\*right;

int val;

node(int x){

val = x;

}

};

typedef struct node node;

The structure is already declared, you can just start initializing nodes

\*/

}

Implement a method for a web service call that receives a collection of n coordinate points and returns

the kth closest point to the origin (0,0). For example, if k=1 the closest point to the origin should be

returned. If k=n the furthest point from the origin should be returned. (Use Divide and Conquer Approach looking for implementation in Java)

nth size of array Push all the non-zero's of a given array to the end of the array. In place

Given a String "abcxrrxabcrr"

Find the first repeated string with minimum 3 character?

Answer is "abc" min 3 characters.

What is the difference between heap and priority queue?

What is its time complexity?

Time Complexity of array(both sorted and unsorted) and doubly linked list(both sorted and unsorted)?

What is the difference between overloading and overriding?

Given an int array sequence:

1

1,1

2,1

1,2,1,1

1,1,1,2,2,1

3,1,2,2,1,1

1,3,1,1,2,2,2,1

1,1,1,3,2,1,3,2,1,1

...

What's the rule of the sequence?

Given an other int array a[], how to determine if a is in the sequence?

Design a Restaurant Reservation system.

Given 2 BSTs. Find the largest common bst between them.

WAP to print k closest points in a plane having n points (2D).

You have a two dimensional array of size m\*n. The

elements in the rows are sorted and every row has

unique elements means( in a row no two elements are same) but

the elements can repeat across the rows.

For example:

you have following 2-D array:

{

2 5 7 9 14 16

3 6 8 10 15 21

4 7 9 15 22 35

7 8 9 22 40 58

}

You are supposed to write an efficient function which

will take upper 2-D array as input and will return a

one-dimensional array with following properties:

a) the 1-D array must contain all the elements of above

2-D array.

b) the 1-D array should not have any repeated elements.

c) all the elements should be in sorted order.

For example:

for the above 2-D array, the output should be:

A [ ] = { 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15, 16, 21, 22, 35,

40, 58 }

Function Prototype should be :

int [ ] MergeAndSort( int[ ][ ] inputArray )

you r given an array and you have to find out the top 3 repeated numbers.

for e.g. RAM[] ={20,8,3,7,8,9,20,6,4,6,20,8,20}

so the output will be : 20 is repeated 4 times , 8 is repeated 3 times, 6 is repeated 2 times .

complexity must be best

differences between integer and int

given stream of integers? find first 100 large numbers

how garbage collector works in java

what is virtual binding?

What is deadlock? example

Given file tokens find common token?

Given an integer, i, find whether it can be expressed as 2^k. Where k< i

Given an array say [9,20,-2,-45,23,5,1], find the minimum positive missing number from the array.

The answer in this case will be 2.

No coding, just was asked to tell how I would do this:

Adding a new part to the webpage that shows recently viewed items.

Questions: What items would you put in the web page?

How would you design the data structure?

How many items should be put on the list?

What would be size in bytes if we store 10 items per user?

Discuss other issues.

Implement a cache that stores a fixed amount of data, provides random access to the elements and is circular (like after completely filling a cache array, overwrite policy is overwriting the first item, then the second item and so on)

Given: A tree in which each node has a pointer to its parent and two nodes.

Print path for each of the two nodes from node to root.

If path for one node is partially same for the second node then only print the part of the path that is not same.

Do zigzag level order traversal of a BST

Find the nodes at d distance from a certain node in a Binary Tree

Find the largest number obtained by rearranging the digits in O(n) time.

Why C language does not support column major array?

Given a BST, with nodes having a parent pointer, a pointer to a node (any node), and a value. Find the path from the given node (pointer), to the node with the given value.

Find the element that appears once

Given an array where every element occurs three times, except one element which occurs only once. Find the element that occurs once.

Expected time complexity is O(n) and O(1) extra space.

Examples:

Input: arr[] = {12, 1, 12, 3, 12, 1, 1, 2, 3, 3}

Output: 2

Total number of steps are given (let N)

at a time you can take one or two steps..

Q total number of ways to reach N.

My Ans: find out total number of non -ve solution to the

equation X + 2Y = N

For every pair of value, find out total number of arrangement. Ex X = x and Y = y

then arrangement is (x+y)!/(x! \* y!).

I think my answer is correct but interviewer was not convince.. may be he is looking some mugged answer ..

Design a system where you can reutrn top 20 queries made in last 24 hours to users.

Think on the scale of Google and Yahoo. How would you store data. What will be your data structures, algorithm to get that data.Describe your assumptions etc.

For simplicity, you can assume that every web server create a log file with query and timestamp.

given arr1 = {5,6,4,2,2} arr2={4,2,2,1}

return common elements {4,2,2}

HashMap h=new HashMap;

h.put(1,tom);

h.put(1,tom)

how hashmap deal with duplicate entries...

"What will happen if two different objects have same hashcode?”

Given a graph and asked to print some order which is BFS..so asked to implement BFS

and each node is of type person

class person{

int ssn;

String name;

}

Given an NxN matrix with unique integers : Find and print positions of all numbers such that it is the biggest in its row and also the smallest in its column.

Write a program that takes an array of numbers, and then prints out all the possible pairs of numbers that sum up to the value N.

E.g., if the array contains the numbers {0, 1, 2, 2, 3, 4, 5} and the target value N is 4, then the output would be (0, 4), (1, 3), (2, 2).

EMP (empno, name, deptno, salary)

DEPT (deptno, name)

Write a SQL query that tells me the average salary by department name.

you have a matrix of N x N filled with 1 or 0. you can move only 1 step either right or 1 step down. you need to count maximum number of "connected 1" in given matrix. for example

0 0 1 1

1 1 1 0

1 0 1 0

0 1 0 1

[Start from top left] maximum no. of 1 is 4

[1,0][1,1][1,2][2,2]

If we have a very big text file which contains millions of lines of code. On every line there is a one word. We also have a random generator which returns a number between 0 and 1. When we have 1 - this will corresponds to the last line of the file. When we have 0 - to the first one. We want an algorithm which tells as the word (the line) which corresponds to the generated number.

Write an algorithm to check if a tree is binary search tree

A soccer league has n matches with A,B,C teams with number of goals scored by each team in each match.If a team wins against another team ,It gets 3 points and lost team gets 0 point.If a tie ,both team gets 1 point.Now how do you frame the ranking of teams.

1) All teams played n matches.

2) A team 1 match,B Team 2 matches, 3 team 3 matches.Like wise it goes.

They looked for coding and data structure techniques.

A log file which has user details(user ID,timestamp) and pages visited in a particular day by that user.The next day -the same kind of log file gets generated.How do you find the probability of users who logged in consecutive days out of the second day - logged in users? The question is simple,but they look for the efficient data structure and time complexity.

Assume there's a website with 8 pages. We are interested in calculating the most frequently visited page sequences of size 3( e.g 1->5->2 ).We are given a log file that has several rows for a particular time period. Each row has following info : time, UserID, page visited

Suggest an optimal algorithm to find the most frequest visited page sequence of size 3.

In a circle there are N number of Gas filling stations which can provide you will gas of capacity C1, C2…CN. Distance between filling station is D1, D2, D3……DN. Considering your vehicle will consume 1 volume of gas per distance. Find the filling station from which you should start the journey so that you will never get short of Gas.

Develop a function which will return either true or false. But at any point of time it should not return more than N number of true (it can be less than N) in last X secs.

Given an initial word and a final word(both same length) suggest an algorithm to find if there were some intermediate steps to convert the initial word to the final word.

The word at every step should be just one character different from the previous word and should be a valid dictionary word.( E.g TAP -> TOP is allowed, since the only difference from previous word is 2nd character and TOP is a valid dictionary word , but TAP -> TOO isn't allowed since 2 characters are different).

Write a function that will return true if a circular singly linked list has duplicate values. For example, given a pointer to a node in the circular singly linked list, \*slist, where the only values each node of this list contains int value, and \*nxt\_pointer. How would you traverse it and what way will allow you to have the best case for time-complexity? How would we know when the circular singly linked list stops?

Array on integer is given

find out next bigger number

Ex {2,5,3,4,61}

Out: 2->5

5->6

3->4

4->6

6->-1 //not possible

1-> -1 //not possible

How to find length of a singly linked list with loops??

write code in java /c for expression 5+4\*(7-15) or have parenthesis in any order .

Given two arrays, write a program to merge them to a new sorted array ? and give test cases..

Write a program that finds whether the two binary trees are mirror image of each other?

write code to find the height of a character which is written on the m\*n Grid. where 1 denote the character on the grid.

The dynamic-set operation UNION takes two disjoint sets S1 and S2 as input, and

it returns a set S = S1 U S2 consisting of all the elements of S1 and S2. The

sets S1 and S2 are usually destroyed by the operation. Show how to support UNION

in O(1) time using a suitable list data structure.

Find the lexicographic rank of a string in the list of permutations of all characters of that string.

Find the first missing number in an array of sorted numbers.

Eg: Input : 4,5,6,7,9,11,14,18,19

Output : 8

Expected complexity : O(log n)

Approach is similar to binary search

Write a function to calculate the difference between the sum of all nodes' value at even and level of a Tree.

Write a function to sort linear Linked list without using extra memory.

Reverse word in the string

Find the least common ancestors for two tree node. Given the parent point.

Given two arrays, array1 and array2

using the rule of array1 to sort array2.

Ex.

array1 = { B, A, C}

array2 = {A, B, A, C, A, B, B, C, A}

output: sortedArray2 = {B,B,B,A,A,A,A,C,C}

What if array2 existed some element not existed in Array1? Can you put it in the end? and sorted by alphabetical? What if array have lower case and upcases letter?

generate permutations of a string without duplicates and without using hashtable to memorize the permutations.

Given two singly linked list, find if they are intersecting. Do this in single iteration. Also find the intersecting node in O(n) time and O(1) space. By intersection I mean intersection by reference not by value

write insert method to insert a node into sorted circular linked list (Sorted based on int value). insert method takes 2 arguments, one is value to be inserted and other is reff to any random node in the sorted circular linked list

An array of building coordinates (x-axis point from origin,height,width) in units were given as an input. Buildings can overlap. We have to give the side view as an answer. Input will be given in sorted order based on x-axis point.

Ex : Draw rectangles based on the given co-ordinates to understand the problem better.

i/p : (5,10,25),(10,20,15),(15,5,5)

o/p : 5R 10U 5R 10U 15R 10D 5R 10D

means draw line 5units Right 10units Up 5 units Right ...

here

R-right, U-up , D- Down

the o/p should be in such a way if we follow that, we should be able to draw side view of those buildings.

Expected better than O(n^2) solution.

Just came from interview. The interviewer asked me this question: copy tree with unlimited number of children with breadth fist search...

given a very large file with words. Reverse the order of all words. For Example: How are you doing today? -->>today? doing you are How I know you may say reverse the whole and reverse each word but, the interviewer said that the file is too big to fit into the memory and the file should be read through a inputStream object, then write the result file to harddisk.

construct a BST given its preorder traversal. solution which i gave :-

make first element of array as node of tree and then if element is less than root and if greater then on right. but i got the answer right for the given example but i am not sure if it was right. can you please suggest me a method to do it.

Give a rectangular matrix(order mxn), each cell having only 0's or 1's, find the largest rectangular sub-matrix with equal number of 0's and 1's in it. O(m^2 n^2) solution possible... More time efficient algorithm required... Is O(mn) possible?

I need to insert an element in a heap, making sure that it does not already exist in it. Can this be done in O(logn) time?

find longest palindrome in a given string, expecting time complexity must be less than O(n^2).

Given a million points (x, y), give a O(n) solution to find 100 points closest to (0, 0).

Find the union of non-overlapping ranges,

e.g; given an array {0,3,1,5,7,9,8,13} where 0 is starting point and 3 is the end point and so on.

The output should be {0,5,7,13}

Given a tree with following special property, develop an algorithm to find the LCA of two input nodes. Only O(1) variables can be used.

property - all nodes have information only about their parents not their children.

// Finds and returns the value of the element that occurs an odd number of times

// in the input array. Examples:

// [1000000] => 1000000

// [2, 2, 2] => 2

// [1, 3, 3] => 1

// [1, 2, 3, 1, 2, 3, 4] => 4

// [1, 2, 3, 2, 1, 2, 3, 2, 1] => 1

// etc.

int findOdd(int[] input, int length) {

}

Here is a Java function API: int [] intersection(int [] a, int [] b). Provide your implementation of this Java function. (Note: it is an easy question. They are looking for code quality.)

There is a sorted array of infinite numbers (can contain duplicates). Given a number. Find the last occurring instance of that number in the array.

Ex., i/p: 12344445566667789...

search number: 6

o/p: 12 (index)

Given: a file with all possible words

Given: word like in Hangman game: H\_L\_

Output: getPossibleSolutions(“H\_L\_”) -> set contains strings “HELL”, “HELP”

given an ASCII string, return the longest substring with unique characters. Ex: dabcade => Ans: bcade.

Stream has 2 methods

char getNextChar()

bool hasNextChar()

Stream is expected to have 1 M characters. Your application cant store them.

Want to find the 1st unique character in the stream

How will you remove duplicate characters in a string

You are given an array which contains coin denominations. e.g. d = {1,2,3,5,8,12,15,21,37,56}. Each of these denominations are in infinite numbers. Write a program to produce the minimum number of coins(with denomination) which sum up to a given value. e.g. 189

output: 3(56)+1(23) = 4

you may produce output as 56->56->56->23

You are given 2 fair dices, whose all 6 faces are blank. You have to fill these faces with 0-9(you can repeat some digits, as there are 12 faces) such that, you should be able to produce all the dates of an English month.

e.g. you should be able to show 01 02 ....09 10.....30 31

Given two binary trees. Write a function to determine whether they are similar or not.

Given a word and special characters print all the combinations of that word along with atmost 2 special characters

Word - test

special characters 0-9,%,#,&,!

Eg: test, test1, test2, te3st, te1!st, test#%, 12test, !te5st, t0est&, etc. etc.

Given a 2-D MxN matrix having each value as difficulty for the block. A frog is starting from a point Matrix[0][0] and will have to reach Matrix[M-1][N-1]. It can jump any step in one go [ 1, 2, ..... M-1] horizontally OR [ 1,2,3,.... N-1] vertically

Each difficulty value is positive. Write code to give path trace for frog.

Two structure to use -

struct node

{

int x;

int y;

struct node \*next;

};

struct path

{

int difficulty;

struct node \*pathlink;

}

Ex matrix - 4X4 matrix

7 9 2 11

13 23 1 3

14 11 20 6

22 44 3 15

Minimum difficulty = 7 (a[0][0])+ 2(a[0][2]) +3(a[3][2])+15(a[3][3]) = 27

Path trace will have = 7->2->3->15

You have a list of 1 million distinct English words. Each word is between 1 to 40 letters long and contains only alphabets, no space or special characters. The list is already sorted alphabetically.

Given a word shorter than 40 letters, find all words in this list that are only 1 letter different from this word, spelling order is not important.

There are 3 types of matches: 1. Swap one letter with another and you have an exact match 2. Remove one letter and you have an exact match and 3 Add one letter and you'll have an exact match. For example: Given the word "coverage", these are valid matches:

1. "converge". Swap 'n' with 'a'

2. "coverages". Remove 's'.

3. "overage". Add a 'c'.

What's the time complexity of your algorithm?

Can your algorithm handle the request to find words that are 2 letters different from the given word?

Given an integer (assume it's smaller than 50), write an algorithm that will generate all possible combinations of integers greater than 1 and they produce a sum equals to this number. The same number can appear more than once in a combination. What's the time complexity of your algorithm?

For example:

<=1 -> {}

2 -> {2},

3->{3},

4->{[4], [2, 2]},

5->{[5], [3, 2]},

6->{[6], [4, 2], [3, 3], [2, 2, 2]}

7->{[7], [5, 2], [4, 3], [3, 2, 2]}

8->{[8], [6, 2], [5, 3], [4, 4], [4, 2, 2], [3, 3, 2], [2, 2, 2, 2]}

....

I have 10 million 10-bit integers to sort, how would you sort them and what's the time complexity?

Follow-on question: Instead of sorting integers, I now have 10 million pairs to sort. Each pair consists of a 10-bit integer and an object, the sort order is determined by the 10-bit integer. Will your original sort algorithm hold or do you need sort it differently?

(A word of advice: Ask as many questions as you want during the interview, but you MUST be quick. Also, don't mention anything until you've thought it through clearly, otherwise you're just inviting more questions. Time is of essence, you're too slow if this question takes you more than 15 minutes to come up with the optimal solution, because remember, you have to leave time for explanations and other questions)

Given an array of integers and a function Arrange(int position), the function takes the position of an element in the array as input and puts this element at the last position and arranges the array. Now the objective is to sort the array using the Arrange function minimum number of times.

In a file stream, you are at unknown position. You have an API to move forward or backward 1 Byte at a time and the pointer points to first bite.

Assumption: Chinese character occupies 4 bytes and English character occupies 2 bytes. Chinese character always starts with first bite value 1 (means first bite of the character out of 4 bytes) and English character with first bite value 0. Any character (Chinese or English) is identified by first bite of fist Byte of the character.

Example: x is 0 or 1

Chinese 1xxxxxxxx|xxxxxxxxx|xxxxxxxxx|xxxxxxxxx

English 0xxxxxxxx|xxxxxxxxx

Problem Statement: You need to develop an algorithm or code (C++) to find out the character that the current pointer pointing to.

Suppose we have a some part of stream like CCECC and pointer at 6th Byte, then the current pointer that it is pointing to Chinese character.

C | C |E | C | C

BBBB|BBBB|BB|BBBB|BBBB

++++ ++

You have a matrix of size (m x n). find submatrix of size (k x k) with maximum possible sum. 0<k<m and 0 <k <n

[HINT] use DP

You are given a binary search tree T of finite (means can fit in memory) size n in which each node N contains

- integer data element

- pointer to left child

- pointer to right child

- pointer to in order successor (which is set null for each node)

Set all in order successor pointers of the given binary search tree.

Given a conteneous input stream of integer.. Find out Maximum N number at any given instance..

You are given with N number of linklist of max size k. Some linklists are merged with each other and some are stand alone. Return all stand alone linkedlist... not just number of stand aone...

Given an 2D array which is row wise sorted and column wise sorted. Search a given element fron the array.

Given two arrays, A and B, both containing integers, find values that appear in both arrays and output them.

I knew the fastest answer to this, which is basically adding array A to a hashmap and then checking if that map contains each element of B, which is an O(n) operation, but uses memory in O(n) as well. The interviewer then asked if I could figure a way of doing this with a complexity of O(n) without using any extra memory, basically just O(1) for memory.

Is this possible? I could not think of a simple quick solution for this on the fly, but I imagine it is possible.

Here is the code I wrote during the interview.

Also, another quick question, is it typical for a phone interviewer to only ask you one question? I think it would be kind of difficult to ask more than one technical question, including coding, in such a short amount of time, i.e. < 1 hour

Dictionary d: boolean isWord(String)

Input: "thisisawesome"

Output:

this is awesome

this is awe some

isWord("this") == true

isWord("esome") == false

Design an algorithm to find the least common ancestor of two nodes in a Binary tree(Note: Its not a binary search Tree)

Node Structure is given as

Given a string, you have remove duplicates from it in O(n) time and O(1) space.

we have streaming numbers, where numbers may repeat for example 1,2,5,11,2,8,2,9 number 2 is repeating. provide nth number where n is the index of number, excluding the repeated number. hence for above example if n is 3 return 8 because 2 is repeating so excluded. best time and space. code in java.

How to find maximum path sum in a binary tree.

The path need not be a top-bottom, can start and end nodes need not be root or leaf, path can start in left/right subtree and end in right/left subtree wrt any node.

Give a BST and a number. we need to find next bigger number in BST.

We have n number of sorted array for fixed length.

Now we have to merge these and need to save finaly result array into given array.

Note- we can't use extra space except the given array.

first non repeating character from a given string

find the sub array of sum K from the the given unsorted array

only pre oder of BST is given need to construct BST in O(n)...O(nlog(n)) is rejected

matrix contains a 1's and 0's find the entities filled with 1's and bounded by 0's ...we need to give the no of such entities exists in the matrix

given a binary search tree, find the any 3 nodes such that sum is K

find the largest BST from the given Binary tree

3)array contains only 0 and 1's need to sort the array such that all zeros at first and 1's later part of the array

2)2D graph, there are two pints (x1, y1) (x2, y2) x1<x2 & y1<y2 and we are supposed to move in positive direction either right or up i.e (x1+1) or (y1+1) only

we need to find the no of paths to reach that point x2, y2

total 3 questions

1)pre fix to post fix

Given two node values for a tree, find LCA. Consider edge cases - (1) If there exists only one value in the tree, return null. (2) If none of the values exist in the tree, return null. (2) Handle ancestors for duplicate values like -

20

/ \

10 50

/ \ / \

10 17 40 60

/ \

5 15

So for above tree if values 5 and 17 are given, there LCA is the upper 10 and not the lower. You can exemplify something I bet :-)

Given a sorted array of size n implemented as ring buffer, so that when array has reached (n-1) index, it will be overwritten from 0th index and likewise it will continue. Given any number, find its index in the array in less than linear time. Interviewer asked to consider the edge cases which I could not quite understand during the interview. I told, the edge cases would be no-element array, array with one element and such.

I will try to state the question. We all know mobile phone's keypad where "2" is mapped to ABC, "3" to DEF and so on. Given any sequence of integers, find all the (matching) combinations in your phone book.

Write an algorithm to print out how many extra duplicates there are in a binary search tree.

input 1:

2

/ \

1 2

output 1:

2 1

input 2:

3

/ \

2 3

/ \ \

1 2 4

/ \

3 4

\

5

\

5

output 2:

2 1

3 2

4 1

5 1

Given:

Node {

int value;

Node left;

Node right;

}

Remove the duplicates in a linked list

I don remember the exact question. I guess it was " Given an array where the first and 2nd haalf of the array are sorted. Print the sorted array"

Online Test: Find the least common ancestor for any two given nodes in a Binary search tree

Given an array of n integers, Display the numbers with even frequency.

Reverse a string. Give complexity.

Implement a stack using two queues (no coding necessary)

Check if a tree is a binary search tree

(screening round)

Implement atoi. What would be your approach converting for string to hex.

(screening round)

Given two sorted arrays, merge them into result array with sorting. Time and Space Complexity.

Given a ternary string, you have to count the total number of contiguous substrings (contigious set of characters), that you can form from this given string such that they comprise of either only one or two different characters.

Please note that a unique substring will be decided by its starting and ending indices. So, a substring 'ab' with starting and ending indices being 1 and 2 respectively should be considered different from a substring 'ab' with starting or ending indices (or both) other than 1 and 2 respectively.

For example:

input ternary string - aabc

output - 8

The above string comprises of the following substrings that have either one or two of the characters - a, a, b, c, aa, ab, bc and aab. So the final answer is a total of eight substrings.

input ternary string - abc

output - 5

The above string comprises of the following substrings that have either one or two of the characters - a, b, c, ab and bc. So the final answer is a total of five substrings.

input ternary string - baaccb

output - 16

The above string comprises of the following substrings that have either one or two of the characters - b, a, a, c, c, b, aa, cc, ba, ac, cb, baa, aac, acc, ccb and aacc. So the final answer is a total of sixteen substrings

Consider there are n matrices. For eg, A, B, C and D are four matrices. Find the groupings of matrices during their product, the operations involved in your choice of grouping is minimal.

For eg, you can group like (AB)CD or (ABC)D or A(BC)D or A(BCD) .... But among these options in which grouping the operations of matrix multiplication will be minimal. Remember in matrix multiplication , multiplication and sum of elements are involved.

For an array of n integers and a number k between 2 and n, give an algorithm to determine if there are k elements that sum to zero. What are the time and space complexity?

In a hall there are many peoples, where some people know each other, one one person who dont know anyone but all other people know him, So we have to find that person.

You have to use this method

boolean knows(personA, personB){}

It will return true if person knows each other other wise false.

Given an array of numbers (integers) find all pythogorean triplets (a^2 + b^2 = c^2). print a,b an c and the indexes.

Prefix Expression to Postfix Expression.

Given two positions in a 2-D matrix, say (x1, y1) and (x2, y2) where x2>=x1 and y2>=y1. Find the total number of distinct paths between (x1, y1) and (x2, y2). You can only move in right direction i.e. positive x direction (+1, 0) or in up direction i.e. positive y direction (0, +1) from any given position.

Example: If the given coordinates are (3,3) and (5,5), the number of distinct paths are 6 : one going through 3,5 ; one going through 5,3 and four going through 4,4.

Given an array containing sequence of bits (0 or 1), you have to sort this array in the ascending order i.e. all 0' in first part of array followed by all 1's. The constraints is that you can swap only the adjacent elements in the array. Find the minimum number of swaps required to sort the given input array.

Example: Given the array (0,0,1,0,1,0,1,1) the minimum number of swaps is 3.

Note: You just need to complete the function given below for this task. The function is given a binary string as input and returns the required answer.

Given the root of a tree and two other nodes n1, n2. Find the distance between n1 and n2

Given a sorted string s1 and an unsorted string s2. Sort s2 based on the sorting algorithm applied in s1.

Ex:

s1 = fghab

s2 = abfmgfghnaixcv

Output: ffgghaabmnixcv

An operation "swap" means removing an element from the array and appending it at the back of the same array. Find the minimum number of "swaps" needed to sort that array.

Eg :- 3124

Output: 2 (3124->1243->1234)

How to do it less than O(n^2) ?

Given an arrangement of balls on 2-D Euclidean plane (i.e a flat surface), you have to assign a color to each ball

such that no two adjacent balls are of the same color. A greedy approach can be used to reduce the number of

colors required.

Question:

Model this as a graph problem. [Hint: Balls become vertices, adjacency relation is modeled by edges, and each

vertex has a unique identification number and a color.]. Write a program that finds the number of colors

required and outputs the balls (unique ids) along with their colors.. Note that to solve this problem, all balls and

their neighbors must be inspected.

Use adjacency lists to represent the graph.

The input to this program is a file containing the number of balls in the first line followed by the list of

adjacencies – one per line: e.g. an input line containing

x,y

denotes that balls x and y are neighbors. Here x and y denote the unique ids of the two balls.

A simple greedy algorithm for this color assignment problem is as follows:

I. Sort all the vertices in the graph on the basis of their degrees [This sorting should be done in-place on

the array of adjacency lists.]. Assume colors are ordered c1, c2, …

II. Let u be the un-colored vertex with the smallest degree. [Break ties in favor of the vertex with the

smaller id]

a. Assign first color ci in the list of colors to u such that

color(u) ç ci where ci != color(vj ) for any vertex vj in the adjacency list of u.

III. Repeat step II until all vertices are colored.

Implement your solution using a Graph ADT that supports the following interfaces:

a) Graph createGraph() : Creates an empty graph.

b) Graph addEdge(Graph g, Vertex v1, Vertex v2): adds an edge from vertex v1 to vertex v2 to the graph g.

If a new vertex is found, then an entry has to be added in the adjacency list

c) Iterator getNeighbors(Graph, Vertex): gets a list of neighbors of the vertex.

d) Graph sortGraphbyDegree(Graph) : sorts the adjacency list based on degree of the vertices. The vertex

with smallest degree will appear first, and the vertex with the largest degree will appear last. If two

vertices are having the same degree, then their order of appearance will not change.

e) Color chooseColor(Graph, Vertex): returns the first color in the list of colors that satisfies the condition

mentioned in step (2) above.

f) int assignColors(Graph) : invokes chooseColor vertex by vertex and stores the chosen color in the

corresponding vertex. This function returns the number of colors used.

g) printGraph(Graph , num\_colors\_used, file) : prints the number of colors used, num\_colors\_used, in the

first line of the output file. It then prints the graph into the file using the following format:

(vertexid,color):v1,v2,v3,vn

where vertexid is the unique id of the vertex, color is the color assigned to the vertex, and v1,v2,…,vn

correspond the unique identification numbers of the vertices that are adjacent to the current vertex.

Data structures Used:

Graph: This is a dynamic array, such that each entry in the array contains a pointer to vertex vi, the degree of vi

and a list of neighbors of vi.

Vertex: Each vertex contains the unique identification number of the vertex, its color.

List: This is for the list of neighbors, such that each entry in the list corresponding to vertex vi consists of a

pointer to vertex vj, ? vj Î neighborhood(vi)

Steps to perform:

1. Write the relevant Header files for an adjacency list representation of Graph

2. Write a driver file that reads an input file containing the edges in the graph and prints the graph after

coloring the balls. This driver uses the adjacency list for graph representation.

a. The driver takes the name of the input file and output file as command line parameters.

b. From the file, find number of nodes involved.

c. For each line in the input file corresponding to an edge

i. add the edge using call to addEdge().

d. The driver must then invoke function sortGraphbyDegree() to sort the vertices in the increasing

order of their degrees.

e. Invoke assignColors to assign colours to each vertex.

f. Print the number of colours used and the resultant graph into the output file using the call to

the function printGraph(). If no output file is mentioned in the command-line, then print the

graph into the screen.

3. Write the code for the functions a - g

A stream of numbers of length not more than M will be given. You don't know the exact length of the stream but are sure that it wont exceed M. At the end of the stream, you have to tell the N/2 th element of the stream, considering that N elements came in the stream. what would be best space complexity with which you can solve this problem

What is a HashMap? What is one advantage of using a HashMap versus a TreeMap?

Given an array of integers, find the mode and the frequency of the mode. If possible, print each number along with its frequency.

In a hash map with objects as keys,

a) what method do you have to overwrite to do this?

b) how would you resolve a collision?

Find all elements in an array that appears 1/k times where k is any number such that 1<k<n. n is the size of array.

There is a given linked list where each node can consist of any number of characters :- For example

a-->bcd-->ef-->g-->f-->ed-->c-->ba.

Now please write a function where the linked list will return true if it is a palindrome .

Like in above example the linked list should return true

If you 15,000 HTML files, find all phone numbers in the files?

Last question and only asked to give general ideas!

Given two numbers, print all prime number between two given numbers.

what is hashtalbe?

How to use hashtable?

What is hash function?

How will you deal with hash space conflict?

Write code to delete every Nth node from double linked list.

You have an array of size n with values ranging from 1 to n. Exactly one number is missed and one number is repeated. Find missing number and Repeated number.

Given:-

1) an array of strings.

2) a directed graph whose each node has a character

(graph may be cyclic)

count the number of occurrences of each string(given in array) in the graph.

int isBST2(struct node\* node) {

return(isBSTUtil(node, INT\_MIN, INT\_MAX));

}

/\*

Returns true if the given tree is a BST and its

values are >= min and <= max.

\*/

int isBSTUtil(struct node\* node, int min, int max) { if (node==NULL) return(true);

// false if this node violates the min/max constraint if (node->data<min || node->data>max) return(false);

// otherwise check the subtrees recursively,

// tightening the min or max constraint

return

isBSTUtil(node->left, min, node->data) &&

isBSTUtil(node->right, node->data+1, max)

);

Change the BSTutil function such that if root==null return false.

After changes the code should tell correctly if binary tree is BSt or not.

Sort an array of characters in linear time complexity (and linear space complexity if that's possible).

Given a list of strings, write an alogirthm that will return a list of sets of

\* permutations.

\*

\* sample input: [abc, cab, ba, b, ba]

\* sample output: [{abc, cab}, {ba}, {b}]

\*/

A queue is implemented using a circular list. If only one pointer is given to which node a pointer p should point such that enqueue and dequeue operation could be performed in o(1).

options are

1) Rear

2)Front

3)Node next to front

4) one more option was there

There are three operations on a stack. push, pop and one extra operation reverse that will reverse the element in stack. Using this we have to implement a queue. so for Enqueue and Dequeue operations how many operations on stack are needed.

options are

1) 3,3

2) 1,1

3)1,3

one more option was there i don't remember

Given a binary tree, where each node has some value.

Print the path with maximum value.

Designa a phone book - basically contact book on phone.

Give data structures and give time complexity to search a phone number.

Ex: search - freeninza and if found in your phone book return the mobile number of user.

Given 2 strings find if they are anagram

Given two strings A and B of different length, find that whether all the characters of A exists in B or not.

For example,

1.

A: 'abcd'

B: 'agbchd' ===> letters of string A exists in B.

2.

A: zzz

B: abz ==> letters of A exists in B.

Given a string with duplicate character. Tell its rank among all its permutations sorted lexicographically. Spcially mentioned that string contains duplicate characters.

Print a binary tree in vertical.

e.g.,

1

/ \

2 3

/ /

4 5

\

6

o/p: 4 2 1 5 3 6

how to sort a single linked list with out using an additional node?

How would you implement PIPE functionality in UNIX systems. Like "tail -f xyz.log | grep amazon"

How would you implement trending topics for twitter. Think at large scale as well.

Please elaborate your assumptions as well.

Design a system where you can reutrn top 20 queries made in last 24 hours to users.

Think on the scale of Google and Yahoo. How would you store data. What will be your data structures, algorithm to get that data.Describe your assumptions etc.

For simplicity, you can assume that every web server create a log file with query and timestamp.

Implement the classes to model two pieces of furniture (Desk and Chair) that can be constructed of one of two kinds of materials (Steel and Oak). The classes representing every piece of furniture must have a method getIgnitionPoint() that returns the integer temperature at which its material will combust. The design must be extensible to allow other pieces of furniture and other materials to be added later. Do not use multiple inheritance to implement the classes.

# Preparation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Subject** | **Titles/Source** | **Prep Days** | **Remarks** |
|  | C | KR  Complete Ref | -  1 |  |
|  | C++ | Complete Ref  TIC++ 1  TIC++ 2  Effective C++  More Effective C++  Effective STL  Modern C++ Design  C++ FAQ (<http://www.parashift.com/c%2B%2B-faq-lite/index.html>)  C++ FAQ Book 2nd Ed.  Stl code | 3  2  -  5  -  3  4  3 |  |
|  | Java | Complete Ref(Core Java )  JDBC, JSP, Serv  Mastering EJB  Hibernate  Struts 1 Struts 2  JSF  Spring  J2EE  J2ME  Web Services  Ajax  Dojo  Flex  Ant | 1  .5  .5 |  |
|  | Algorithm | Sahni ((1,2,3,4),(5,6,7),(8,11))  Coreman(2,4,6,7,8,9), (10,11,12,13,14),  (15,16,18,19,20,21),  (22,23,24,25,26),  (27,31,32,34)  Wikipedia | 3  2 |  |
|  | Data Structures | Tanenbaum | 2 |  |
|  | Design Patterns | GOF | 3 |  |
|  | Operating System | Galvin | 2 |  |
|  | Networks | Tanenbaum | 2 |  |
|  | Mobile Computing | Notes  Book | 1  2 |  |
|  | DataBase | Navathe  Sudarshan  SQL  MySQL | 2  1 |  |
|  | Data mining | Notes  Book | -  - |  |
|  | Unix | Shell scripting (Das)  Unix programming (Kernighan) | 1  1 |  |
|  | Advanced Unix Programming | Stevens | 1 |  |
|  | Pthread | Pthread premier  Cpp\_Concurrency |  |  |
|  | Perl |  |  |  |
|  | PHP |  |  |  |
|  | Digital Electronics |  |  |  |
|  | Puzzles | [Put links and sources] |  |  |
|  | Quanta | RS Agrawal |  |  |
|  | Hackers Delight |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Misc topics | Programming Pearls  Programming interviews exposed  Smart pointer(auto\_ptr)  garbage collection |  |  |
|  | Problems collected on net |  |  |  |

Linked list (Stack, queue, Double linked list)

Binary Trees

Graphs

Pointer & memory

Tree list recursion

books/DS, C, C++, Aptitude, UNIX, RDBMS, SQL, CN, OS.pdf

F:\disk\Literature\ebooks\fromBestin\books\problems.pdf

Basic\_Must\_Read/BinaryTrees.pdf

Basic\_Must\_Read/books/C Programming - Just the FAQs.pdf

Basic\_Must\_Read/books/CPlusPlus\_FAW\_2nd\_Edition.pdf

Basic\_Must\_Read/Collected by CS MTechs/Algos/Divide and Conquer.pdf

Basic\_Must\_Read/Collected by CS MTechs/Algos/dp\_s04.pdf

Basic\_Must\_Read/Collected by CS MTechs/Algos/dynproghw.pdf

Basic\_Must\_Read/Collected by CS MTechs/c/C Traps and Pitfalls.pdf

Basic\_Must\_Read/Collected by CS MTechs/c/c\_interview\_questions\_part1.pdf

Basic\_Must\_Read/Collected by CS MTechs/c/e\_fpt.pdf

Basic\_Must\_Read/Collected by CS MTechs/ds/heaps.pdf

Basic\_Must\_Read/Collected by CS MTechs/ds/Linked Lists.pdf

Basic\_Must\_Read/Collected by CS MTechs/ds/linkedlistproblems.pdf

Basic\_Must\_Read/Collected by CS MTechs/Links/Links.txt

Basic\_Must\_Read/LinkedListBasics.pdf

Basic\_Must\_Read/LinkedListProblems.pdf

Basic\_Must\_Read/PointersAndMemory.pdf

Basic\_Must\_Read/SQL.doc

Basic\_Must\_Read/SQL\_ans.doc

Basic\_Must\_Read/Technical+Aptitude+Questions+eBook.pdf

Basic\_Must\_Read/TreeListRecursion.pdf

# C++ Notes

## Complete Reference Notes

Non-static member variables can’t have initializer.

No member can be an object of the class that is being declared

No member can be auto, extern, or register.

**Unions**

Unions may contain member functions and variables.

They may include ctors and dtors.

No inheritance

Unions can’t be base classes

Can’t have virtual member functions

No static variables

A reference member can’t be used

*A union can’t have as a member any object that overloads the = operator*

*An object can’t be a member of a union if the objet has an explicit constructor or destructor function*

**Anonymous unions**

No objects can be created

Only data, no functions

No private or protected data

Global anonymous unions must be specified as static

**Friend functions**

A **friend** function has access to all **private** and **protected** members of the class for which it is a **friend**

Useful when:

Overloading certain type of operators

Friend functions make the creation of some type of IO functions easier**\*\*\*\*More explanation**

Two or more classes may contain members that are interrelated to other parts of your program.

A derived class doesn’t inherit friend functions

A friend function may not have a storage class specifier, i.e. they may not be declared as static or extern.

A **friend** of one class may be a member of another

**Friend classes**

When one class is a friend of another, it only has access to names defined within the other class. It doesn’t inherit the other class.

Uses**\*\*\*\*More explanation**

**Inline functions**

Constructors and destructors can be inlined.

**Static members**

**Static data member**

Static data members are only **declared** in the class. We must provide a global definition for it elsewhere outside the class. This is done by redeclaring the **static** variable using the scope resolution operator to identify the class to which it belongs

Static member variable exists even before an object is created.

Count the number of objects that are in existence.

**Static member functions**

They can only refer to other static members of class

A static member function doesn’t have this pointer

There can’t be static and non static versions of same function

A static member function may not be virtual

They can’t be declared as const or volatile

**Scope resolution operator**

To allow access to a name in the enclosing scope that is hidden by a local declaration of the same name

**Nested classes and Local classes**

Nested classes: It is possible to define one class within another

Local class: A class defined within a function.

All member functions must be defined within the class declaration. The local class may not use or access local variables of the function in which it is declared (except that a local class has access to **static** local variables declared within the function or those declared as **extern**). It may access type names and enumerators defined by the enclosing function, however. No **static** variables may be declared inside a local class.

**this pointer**

used in operator overloading

A base class pointer can also be used as a pointer to an object of any class derived from that base

You can access only base class members

You can do casting to access all members of derived class object

The use of base pointer to derived types is useful when creating runtime polymorphism through the use of virtual functions

Pointers to class member ( .\* , ->\* )

int cl::\*data; //not specific to any object

data = &cl::ival; //offset

**References**

There are three ways that a reference can be used: as a function parameter, as a function return value, or as a stand-alone reference.

**Reference parameters**

void swap(int &a, int &b)

**Passing references to objects**

void neg(cl &o) {//def o.i=1;}

**Returning references**

Functions to be used on left of =

char& Replace(int i);

replace(5)=’x’;

When returning references the object being referred should not go out of scope after the function terminates

**Independent references**

All independent references must be initialized when they are created.

You can’t ref another ref

You can’t create array of references

You can’t create a pointer to a ref

A ref variable must be initialized when it is declared unless it’s a member of a class, a func parameter, or a return value.

A base class reference can be used to refer to an object of a derived class

**New and delete**

p\_var = new type; int\* p = new int;

initializing allocated memory: p\_var = new type(init); int\* p = new int(10);

allocating arrays: p\_var=new type[size];

Arrays can’t be initialized using new.

Allocating objects: class\_ptr = new ClassName;

placement new: useful for overloading new

p\_var=new (location)type;

**copy constructor**

classname (const *classname* &*o*) {

*// body of constructor*

}

used when one object initializes other

When one object explicitly initializes another, such as in a declaration: myClass x = y;

When a copy of an object is made to be passed to a function : func(y);

When a temporary object is generated (most commonly, as a return value): y = func();

Default arguments

int f(int a, int b=1);

**Function overloading and ambiguity**

1. myFunc(float i);
2. myFunc(double i);

myFunc(10.1); -- > B

myFunct(10);🡪 ??

void f(int x);

void f(int &x);

f(2);🡪ambigous

**Operator overloading**

Member functions

non-member functions (friends almost)

ret-type className::operator#(args){

}

**Prefix and postfix ++ & --**

type operator++() 🡪prefix

type operator++(int x)🡪postfix

You cannot alter the precedence of an operator. You cannot change the number of operands that an operator takes. Except for function call operator “()”, operator functions can’t have default arguments

. , :: , .\* , ? can’t be overloaded.

Except for = operator, operator functions are inherited by derived classes.

**Friend functions for overloading**

=, (), [] , -> can’t be overloaded using friend functions

When overloading ++/--, you need to use a reference parameter when using a friend function

type operator++(className &ob)

{}

type operator++(className &ob, int x) // postfix

{}

Friend operator functions add flexibility

Ob + 100;

100 + Ob; // valid with friend functions.

**New & Delete overloading**

void \*operator new(size\_t size)

{

//perform allocation

// return pointer to allocated memory

}

void operator delete(void \*p)

{

//free(p);

}

void \*operator new[](size\_t size)

{}

void operator delete[](void\* p)

{}

// Nothrow version of new.

void \*operator new(size\_t size, const nothrow\_t &n)

{

// Perform allocation.

if(success) return pointer\_to\_memory;

else return 0;

}

// Nothrow version of new for arrays.

void \*operator new[](size\_t size, const nothrow\_t &n)

{

// Perform allocation.

if(success) return pointer\_to\_memory;

else return 0;

}

**Overloading [] , () , ->**

type className::operator[] (int i)

{}

To use [] on both left and right side of assignment make return type as a reference

type& operator[] (int i)

{}

**Overloading ()**

type operator() (arglist);

When overloading (), you can use any type of parameters and return any type of value.

**Overloading –>**

*class member access* operator

className\* operator->()

*object->element;*

Here, *object* is the object that activates the call. The **operator–>()** function must return a pointer to an object of the class that **operator–>()** operates upon. The *element* must be some member accessible within the object.

**Overloading ,(comma) operator**

Binary operator.

**Inheritance**

Public🡪public remains public, protected remains protected

Private🡪everything become private

Protected🡪public becomes protected, protected remains protected

(the access specifier is **private** by default if the derived class is a **class**. If the derived class is a **struct**, then **public** is the default in the absence of an explicit access specifier.)

**Constructors, Destructors and Inheritance**

Constructors are called in order of derivation, left to right, as specified in **derived**'s inheritance list. Destructors are called in reverse order, right to left.

Passing parameters to base class constructors

*derived-constructor(arg-list) : base1(arg-list),*

*base2(arg-list),*

*// ...*

*baseN(arg-list)*

{

// *body of derived constructor*

}

**Granting access**

When a base class is inherited as private, all public and protected members of that class become private members of the derived class. However, in certain circumstances, you may want to restore one or more inherited members to their original access specification.

In Standard C++, we have two ways to accomplish this.

First, you can use a **using** statement.: *using base-class::member;*

The second way to restore an inherited member's access specification is to employ an **access declaration** within the derived class. *base-class::member;*

**Virtual base classes**

B:i

D1:j D2:k

D3:i?? which i

There are two ways to remedy the preceding program.

1. The first is to apply the scope resolution operator to i and manually select one i.

2. When two or more objects are derived from a common base class, you can prevent multiple copies of the base class from being present in an object derived from those objects by declaring the base class as **virtual** when it is inherited.

**Polymorphism**

A *virtual function* is a member function that is declared within a base class and redefined by a derived class.

When a base pointer points to a derived object that contains a virtual function, C++ determines which version of that function to call based upon *the type of object pointed to* by the pointer. The same effect applies to base-class references.

Virtual functions must be non-static members of a class

They can’t be friends

Constructors can’t be virtual; but destructors can

When a virtual function is inherited, its virtual nature is also inherited

**Pure virtual functions**

virtual type func(param) = 0;

You can’t create objects of abstract classes, but we can create references and pointers of abstract classes.

**Templates**

**Template functions**

template<class Ttype> ret-type Func (arglist)

{

}

***explicit specialization***

template<> ret-type Func <SpecializedType>(arglist)

{

}

**Generic classes**

template <class Ttype>

class className

{

};

Creating object of a template class 🡪 className<type> ob;

Defining member functions of template class

template<class Ttype> ret-type ClassName<Ttype>::member-Func(arglist)

{

}

**Non-type arguments with Generic classes**

-int, pointer or references

Non-type parameters are restricted to integers, pointers, or references. Other types, such as **float**, are not allowed. The arguments that you pass to a non-type parameter must consist of either an integer constant, or a pointer or reference to a global function or object. Thus, non-type parameters should themselves be thought of as constants, since their values cannot be changed.

**Using default arguments with template classes**

template<class X=int> class MyClass

{ };

It is also permissible for non-type arguments to take default arguments.

**Explicit Class Specializations**

template <> class myclass<int>

**typename**

The second use of typename is to inform the compiler that a name used in a template declaration is a type name rather than an object name. For example,

typename X::Name someObject;

ensures that X::Name is treated as a type name.

**export**

The **export** keyword can precede a template declaration. It allows other files to use a template declared in a different file by specifying only its declaration rather than duplicating its entire definition.

**Exception Handling**

**try**, **catch**, and **throw**.

Throwing an unhandled exception causes the standard library function **terminate()** to be invoked.

**Restricting Exceptions**

*ret-type func-name*(*arg-list*) throw(*type-list*)

{

// ...

}

Attempting to throw an exception that is not supported by a function will cause the standard library function **unexpected()** to be called.

If you wish to rethrow an expression from within an exception handler, you may do so by calling **throw**, by itself, with no exception

**Setting the Terminate and Unexpected Handlers**

terminate\_handler set\_terminate(terminate\_handler *newhandler*) throw( );

unexpected\_handler set\_unexpected(unexpected\_handler *newhandler*) throw( );

Both **set\_terminate()** and **set\_unexpected()** require the header **<exception>**.

**The exception and bad\_exception Classes**

When a function supplied by the C++ standard library throws an exception, it will be an object derived from the base class **exception**. An object of the class **bad\_exception** can be thrown by the unexpected handler.

**C++ Stream classes**

|  |  |  |
| --- | --- | --- |
| **Template Class** | **Characterbased Class** | **Wide-Characterbased Class** |
| basic\_streambuf | streambuf | wstreambuf |
| basic\_ios | ios | wios |
| basic\_istream | istream | wistream |
| basic\_ostream | ostream | wostream |
| basic\_iostream | iostream | wiostream |
| basic\_fstream | fstream | wfstream |
| basic\_ifstream | ifstream | wifstream |
| basic\_ofstream | ofstream | wofstream |

**C++’s predefined streams**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stream** | **wchar\_t streams** | **Meaning** | **Default Device** |
| cin | win | Standard input | Keyboard |
| cout | wout | Standard output | Screen |
| cerr | werr | Standard error | output Screen |
| clog | wlog | Buffered version of cerr | Screen |

**typeid**

#include <typeinfo>

typeid(object);

typeid returns a reference to an object of type type\_info. ==, != , before(), name()

bad\_typeid exception is thrown when ptr is null

typeid can only be applied to polymorphic classes

typeid(type\_name)

typeid can be applied to template classes

**Casting operators**

dynamic\_cast

dynamic\_cast<target-type> (expr);

The target type must be a pointer or reference type and the expression being cast must evaluate to a pointer or reference

D \*dp = dynamic\_cast<D\*>(bp);

Dynamic\_cast can be used with template classes

Const\_cast

Const\_cast<type>(expr);

To remove constness

Const int \*val;

int \*p = const\_cast<int \*> (val);

\*p=\*val \* \*val;

Const\_cast can also be used to cast away const-ness from a const reference

Const int &val;

Const\_cast<int&>(val) = val \* val;

Sataic\_cast

Non-polymorphic cast ; similar to simple casting.

Static\_cast<type>(expr);

Reinterpret\_cast

Converts one type into fundamentally different type.

Example pointer to integer

**Namespaces**

**Conversion functions**

operator type()

{return value;}

No arguments

**Const member functions and mutable**

Member functions declared as const can’t modify this pointer. You can’t modify the object calling the function

Const functions can only call other const functions

If you want some members of the class to be modifiable by const functions then declare them as mutable

**Volatile member functions**

this is treated as volatile

int f() volatile;

**Explicit constructors**

To not allow the automatic conversions of on argument constructors

MyClass x =10; // converts to MyClass(10);

But if the constructor is explicit them MyClass x = 10; is not valid

Asm

Assembly in C

asm(“cp code”);

## Simulating final class in C++

template <class T>

class MarkFinal {

private:

MarkFinal(){}

friend T;

};

class Final: virtual MarkFinal<Final>{

public:

Final(){}

};

class Derived//: public Final // Compiler error

{

public:

Derived(){}

};

int main()

{

Final f;

}

Sol 2

/\* A program with compilation error to demonstrate that Final class cannot

be inherited \*/

#include<iostream>

using namespace std;

class Final; // The class to be made final

class MakeFinal // used to make the Final class final

{

private:

MakeFinal() { cout << "MakFinal constructor" << endl; }

friend class Final;

};

class Final : virtual MakeFinal

{

public:

Final() { cout << "Final constructor" << endl; }

};

class Derived : public Final // Compiler error

{

public:

Derived() { cout << "Derived constructor" << endl; }

};

int main(int argc, char \*argv[])

{

Derived d;

return 0;

}

# Effective C++

## Prefer consts, enums, and inlines to #defines

Replace #define with consts

1. #define ASPECT\_RATIO 1.653 -> const double AspectRatio = 1.653;

**Two special cases**

1. The first is **defining constant pointers.** Because constant definitions are typically put in header files, it's important that the pointer be declared const, usually in addition to what the pointer points to

**const char \* const authorName = "Scott Meyers";**

**const std::string authorName("Scott Meyers");**

2. The second special case concerns **class-specific constants**

To limit the scope of a constant to a class, you must make it a member, and to ensure there's at most one copy of the constant, you must make it a static member:

class GamePlayer {

private:

**static const int NumTurns = 5; // constant declaration**

**int scores[NumTurns]; // use of constant**

...

};

What you see above is a declaration for NumTurns, not a definition. Usually, C++ requires that you provide a definition for anything you use, **but class-specific constants that are static and of integral type (e.g., integers, chars, bools) are an exception**. **As long as you don't take their address, you can declare them and use them without providing a definition. If you do take the address of a class constant**, or if your compiler incorrectly insists on a definition even if you don't take the address, you provide a separate definition like this:

**const int GamePlayer::NumTurns;**

You put this in an **implementation file,** not a header file

**Older compilers**

class CostEstimate {

private:

**static const double FudgeFactor;** // declaration of static class

... // constant; goes in header file

};

const double // definition of static class

**CostEstimate::FudgeFactor = 1.35;**

Enum Hack:

When you need the value of a class constant during compilation of the class

class GamePlayer {

private:

**enum { NumTurns = 5 };** // "the enum hack" — makes

// NumTurns a symbolic name for 5

int scores[**NumTurns**]; // fine

};

**Sometimes #defines is a must:**

1. When we have a non-integral constant, such that we will have to give the definition of the static variable outside the class, and we want to use this static constant inside the class.

class InterstCalc

{

static const double rate;

interstCalc()

{

rate\*time\*prinicpal;

}

}

**For function-like macros, prefer inline functions to #defines.**

#define CALL\_WITH\_MAX(a, b) f((a) > (b) ? (a) : (b))

template<typename T> // because we don't

inline void callWithMax(const T& a, const T& b) // know what T is, we

{ // pass by reference-to-

f(a > b ? a : b); // const — see Item 20

}

## Use const whenever possible

Outside of classes, you can use it for constants at **global or namespace scope**, as well as for **objects declared static at file, function, or block scope**. Inside classes, you can use it for both **static and non-static data members**. For pointers, you can specify whether the **pointer itself is const, the data it points to is const, both, or neither**.

char greeting[] = "Hello";

char \*p = greeting, ; // non-const pointer

// non-const data

const char \*p = greeting; // non-const pointer,

// const data

char \* const p = greeting; // const pointer,

// non-const data

const char \* const p = greeting; // const pointer,

// const data

If the word const appears to the left of the asterisk, what's pointed to is constant; if the word const appears to the right of the asterisk, the pointer itself is constant; if const appears on both sides, both are constant.

**const\_iterators**

STL iterators are modeled on pointers, so an iterator acts much like a T\* pointer. Declaring an iterator const is like declaring a pointer const (i.e., declaring a T\* const pointer): the iterator isn't allowed to point to something different, but the thing it points to may be modified. If you want an iterator that points to something that can't be modified (i.e., the STL analogue of a const T\* pointer), you want a const\_iterator:

std::vector<int> vec;

...

**const** std::vector<int>::iterator iter = // iter acts like a T\* const

vec.begin();

\*iter = 10; // OK, changes what iter points to

++iter; // error! iter is const

std::vector<int>::**const\_iterator** cIter = //cIter acts like a const T\*

vec.begin();

\*cIter = 10; // error! \*cIter is const

++cIter; // fine, changes cIter

**functions returning const objects**

**const** Rational operator\*(const Rational& lhs, const Rational& rhs);

**(a \* b) = c;** //Illegal as \* returns const type

**const Member Functions**

The purpose of const on member functions is to identify which member functions may be **invoked on const objects**.

class TextBlock {

public:

...

**const** char& operator[](std::size\_t position) **const** // operator[] for

{ return text[position]; } // const objects

char& operator[](std::size\_t position) // operator[] for

{ return text[position]; } // non-const objects

private:

std::string text;

};

TextBlock's operator[]s can be used like this:

TextBlock tb("Hello");

std::cout << tb[0]; // calls non-const

// TextBlock::operator[]

**const TextBlock ctb("World");**

**std::cout << ctb[0]; // calls const TextBlock::operator[]**

**Bitwise const and logical const**

bitwise is provided by C++: a member function is const if and only if it doesn't modify any of the object's data members

logical is programmers responsibility: a const member function might modify some of the bits in the object on which it's invoked, but only in ways that clients cannot detect.

mutable : if you want to change a data member's value in const function.

**To avoid duplication** call **const version from non-const version** and cast away the constness from return type. To call the const version, cast the \*this to const type.

Should not call non-const from const.

class TextBlock {

public:

...

const char& operator[](std::size\_t position) const // same as before

{

...

return text[position];

}

char& operator[](std::size\_t position) // now just calls const op[]

{

return

**const\_cast<char&>**( // cast away const on

// op[]'s return type;

**static\_cast<const TextBlock&>**(\*this) // add const to \*this's type;

[position] // call const version of op[]

);

}

...

};

**const can be used with pointers, references, iterators, func params, return types, local variables, member functions.**

## Item 4: Make sure that objects are initialized before they're used

* Prefer member initialization list
* const and references are to be initialized in member initialization list
* Order of initialization of members of a class: Base classes are initialized before derived classes, and
* within a class, data members are initialized in the order in which they are declared
* **static objects**: global, namespace scope, staic in class, static in function, static at file scope
* Problem of the order of initialization of non-local static objects defined in different translation units

class FileSystem { // from your library

public:

...

std::size\_t numDisks() const; // one of many member functions

...

};

**extern FileSystem tfs;** // object for clients to use;

// "tfs" = "the file system"

class Directory { // created by library client

public:

Directory( params );

...

};

Directory::Directory( params )

{

...

std::size\_t disks = **tfs.numDisks();** // use the tfs object

...

}

**Directory tempDir( params );** // directory for temporary files

Unless tfs is initialized before tempDir, tempDir's constructor will attempt to use tfs before it's been initialized.

Move each **non-local static object into its own function, where it's declared static**. These functions return references to the objects they contain. Clients then call the functions instead of referring to the objects. In other words, non-local static objects are replaced with local static objects.

class FileSystem { ... }; // as before

**FileSystem& tfs() // this replaces the tfs object; it could be**

**{ // static in the FileSystem class**

**static FileSystem fs; // define and initialize a local static object**

**return fs; // return a reference to it**

**}**

class Directory { ... }; // as before

Directory::Directory( params ) // as before, except references to tfs are

{ // now to tfs()

...

std::size\_t disks = **tfs().**numDisks();

...

}

**Directory& tempDir() // this replaces the tempDir object; it**

**{ // could be static in the Directory class**

**static Directory td; // define/initialize local static object**

**return td; // return reference to it**

**}**

## Know what functions C++ silently writes and calls

Compilers may implicitly generate a class's default constructor, copy constructor, copy assignment operator, and destructor.

class Empty {

public:

**Empty() { ... } // default constructor**

**Empty(const Empty& rhs) { ... } // copy constructor**

**~Empty() { ... } // destructor — see below**

**// for whether it's virtual**

**Empty& operator=(const Empty& rhs) { ... } // copy assignment operator**

};

These functions are generated only if they are needed.

* All default generated functions are inline and public
* Default Dtor is non-virtual, if class is not derived from a Base class which has virtual function.
* Copy Ctor and Copy assignment: The compiler-generated versions simply copy each non-static data member of the source object over to the target object.
* [Imp] If your class has references or const members and the resulting code for copy assignment becomes illegal the compiler will not generate the default implementation for operator=().

template<class T>

class NamedObject {

public:

// this ctor no longer takes a const name, because nameValue

// is now a reference-to-non-const string. The char\* constructor

// is gone, because we must have a string to refer to.

NamedObject(std::string& name, const T& value);

... // as above, assume no

// operator= is declared

private:

**std::string& nameValue; // this is now a reference**

**const T objectValue; // this is now const**

};

std::string newDog("Persephone");

std::string oldDog("Satch");

NamedObject<int> p(newDog, 2);

NamedObject<int> s(oldDog, 36);

**p = s;** // what should happen to

// the data members in p?

As p.nameValue is a reference you cannot make it refer to the s.nameValue, because references can’t be changed.

Or we can modify the string to which p.nameValue points, but this also affects other objects that hold pointers or references to that string, i.e., objects not directly involved in the assignment?

Same dilemma applies for const objects.

## Explicitly disallow the use of compiler-generated functions you do not want

* To disallow functionality automatically provided by compilers, declare the corresponding **member functions private** and give no implementations.

class HomeForSale {

public:

...

private:

**...**

**HomeForSale(const HomeForSale&); // declarations only**

**HomeForSale& operator=(const HomeForSale&);**

};

If somebody from outside the class will call these functions compiler error will come.

But if the member or friend function call these functions linker error will come.

* Using a base class like Boost’s noncopyable is one way to do this.

class noncopyable {

protected: // allow construction

noncopyable () {} // and destruction of

~ noncopyable () {} // derived objects...

private:

noncopyable (const noncopyable &); // ...but prevent copying

noncopyable & operator=(const noncopyable &);

};

To keep HomeForSale objects from being copied, all we have to do now is inherit from noncopyable:

class HomeForSale: private noncopyable { // class no longer

... // declares copy ctor or

}; // copy assign. Operator

* Here we will get compiler warning even for member function or friend function doing assignments.

## Declare destructors virtual in polymorphic base classes

Polymorphic base classes should declare virtual destructors. If a class has any virtual functions, it should have a virtual destructor.

Classes not designed to be base classes or not designed to be used polymorphically should not declare virtual destructors.

# Important Data structures

## Tries

a trie, also called digital tree or prefix tree, is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings.

A trie (from retrieval), is a multi-way tree structure useful for storing strings over an alphabet. It has been used to store large dictionaries of English (say) words in spelling-checking programs and in natural-language "understanding" programs

Replacement for hash.

dictionary

public class MinimalExample{

private interface Node {

public static final Node EMPTY\_NODE = new Node() {

@Override public String getValue() { return ""; }

@Override public boolean containsChildValue(char c) { return false; }

@Override public Node getChild(char c) { return this; }

};

public String getValue();

public boolean containsChildValue(char c);

public Node getChild(char c);

}

public Node findValue(Node startNode, String value) {

Node current = startNode;

for (char c : value.toCharArray()) {

if (current.containsChildValue(c)) {

current = current.getChild(c);

} else {

current = Node.EMPTY\_NODE;

break;

}

}

return current;

}

}

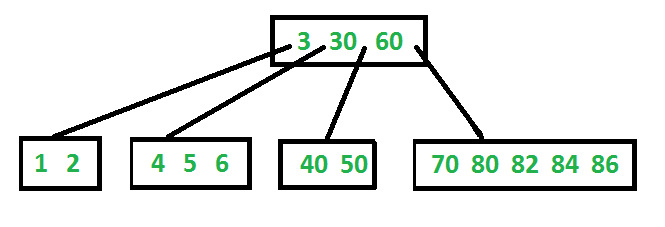
## Radix Trees

a radix tree (also patricia trie or radix trie or compact prefix tree) is a space-optimized trie data structure where each node with only one child is merged with its child. The result is that every internal node has at least two children. Unlike in regular tries, edges can be labeled with sequences of elements as well as single elements. This makes them much more efficient for small sets (especially if the strings are long) and for sets of strings that share long prefixes.

## B-Tree

B-Tree is a self-balancing search tree. In most of the other self-balancing search trees (like [AVL](http://www.geeksforgeeks.org/avl-tree-set-1-insertion/) and Red Black Trees), it is assumed that everything is in main memory. To understand use of B-Trees, we must think of huge amount of data that cannot fit in main memory. When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to main memory access time. The main idea of using B-Trees is to reduce the number of disk accesses. Most of the tree operations (search, insert, delete, max, min, etc.) require O(h) disk accesses where h is height of the tree. B-tree is a fat tree. Height of B-Trees is kept low by putting maximum possible keys in a B-Tree node. Generally, a B-Tree node size is kept equal to the disk block size. Since h is low for B-Tree, total disk accesses for most of the operations are reduced significantly compared to balanced Binary Search Trees like AVL Tree, Red Black Tree, ..etc.

**Properties of B-Tree**  
**1)** All leaves are at same level.  
**2)** A B-Tree is defined by the term minimum degree ‘t’. The value of t depends upon disk block size.  
**3)** Every node except root must contain at least t-1 keys. Root may contain minimum 1 key.  
**4)** All nodes (including root) may contain at most 2t – 1 keys.  
**5)** Number of children of a node is equal to the number of keys in it plus 1.  
**6)** All keys of a node are sorted in increasing order. The child between two keys k1 and k2 contains all keys in range from k1 and k2.  
**7)** B-Tree grows and shrinks from root which is unlike Binary Search Tree. Binary Search Trees grow downward and also shrink from downward.  
**8)** Like other balanced Binary Search Trees, time complexity to search, insert and delete is O(Logn).

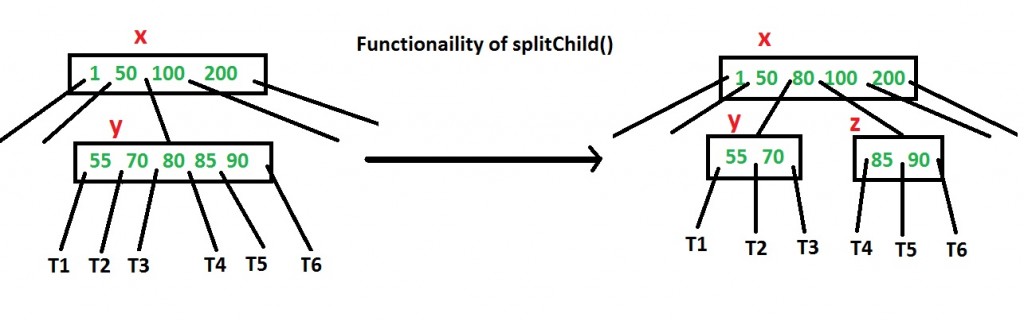
Following is an example B-Tree of minimum degree 3. Note that in practical B-Trees, the value of minimum degree is much more than 3.  
[](http://www.geeksforgeeks.org/wp-content/uploads/BTreeIntro1.png)

**Search**  
Search is similar to search in Binary Search Tree. Let the key to be searched be k. We start from root and recursively traverse down. For every visited non-leaf node, if the node has key, we simply return the node. Otherwise we recur down to the appropriate child (The child which is just before the first greater key) of the node. If we reach a leaf node and don’t find k in the leaf node, we return NULL.

**Traverse**  
Traversal is also similar to Inorder traversal of Binary Tree. We start from the leftmost child, recursively print the leftmost child, then repeat the same process for remaining children and keys. In the end, recursively print the rightmost child.

**Insert**

A new key is always inserted at leaf node. Let the key to be inserted be k. Like BST, we start from root and traverse down till we reach a leaf node. Once we reach a leaf node, we insert the key in that leaf node. Unlike BSTs, we have a predefined range on number of keys that a node can contain. So before inserting a key to node, we make sure that the node has extra space.  
How to make sure that a node has space available for key before the key is inserted? We use an operation called splitChild() that is used to split a child of a node. See the following diagram to understand split. In the following diagram, child y of x is being split into two nodes y and z. Note that the splitChild operation moves a key up and this is the reason B-Trees grow up unlike BSTs which grow down.

[](http://www.geeksforgeeks.org/wp-content/uploads/BTreeSplit.jpg)

As discussed above, to insert a new key, we go down from root to leaf. Before traversing down to a node, we first check if the node is full. If the node is full, we split it to create space. Following is complete algorithm.

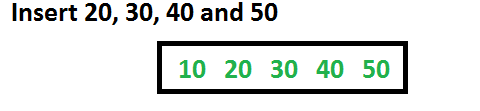
**Insertion**  
**1)** Initialize x as root.  
**2)** While x is not leaf, do following  
..**a)** Find the child of x that is going to to be traversed next. Let the child be y.  
..**b)** If y is not full, change x to point to y.  
..**c)** If y is full, split it and change x to point to one of the two parts of y. If k is smaller than mid key in y, then set x as first part of y. Else second part of y. When we split y, we move a key from y to its parent x.  
**3)** The loop in step 2 stops when x is leaf. x must have space for 1 extra key as we have been splitting all nodes in advance. So simply insert k to x.

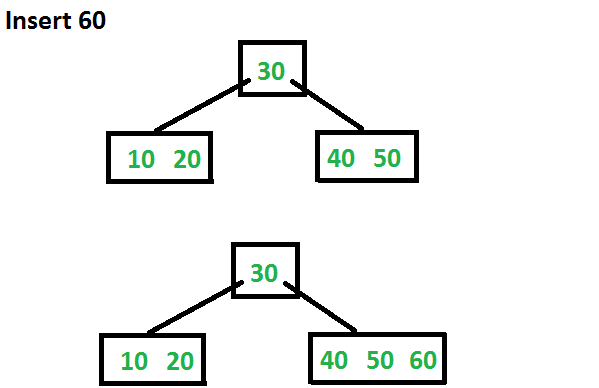
Note that the algorithm follows the Cormen book. It is actually a proactive insertion algorithm where before going down to a node, we split it if it is full. The advantage of splitting before is, we never traverse a node twice. If we don’t split a node before going down to it and split it only if new key is inserted (reactive), we may end up traversing all nodes again from leaf to root. This happens in cases when all nodes on the path from root to leaf are full. So when we come to the leaf node, we split it and move a key up. Moving a key up will cause a split in parent node (because parent was already full). This cascading effect never happens in this proactive insertion algorithm. There is a disadvantage of this proactive insertion though, we may do unnecessary splits.

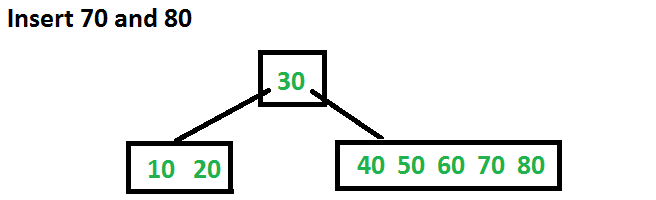
Let us understand the algorithm with an example tree of minimum degree ‘t’ as 3 and a sequence of integers 10, 20, 30, 40, 50, 60, 70, 80 and 90 in an initially empty B-Tree.

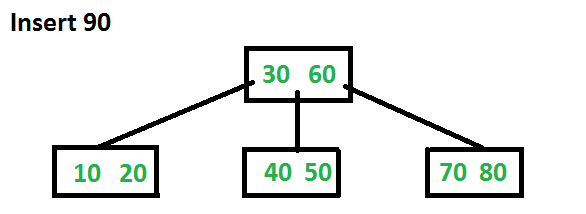
Initially root is NULL. Let us first insert 10.  
[](http://www.geeksforgeeks.org/wp-content/uploads/Btree1.png)

Let us now insert 20, 30, 40 and 50. They all will be inserted in root because maximum number of keys a node can accommodate is 2\*t – 1 which is 5.

[](http://www.geeksforgeeks.org/wp-content/uploads/BTree2Ins.png)

Let us now insert 60. Since root node is full, it will first split into two, then 60 will be inserted into the appropriate child.  
[](http://www.geeksforgeeks.org/wp-content/uploads/BTreeIns3.png)

Let us now insert 70 and 80. These new keys will be inserted into the appropriate leaf without any split.  
[](http://www.geeksforgeeks.org/wp-content/uploads/BTreeIns4.png)

Let us now insert 90. This insertion will cause a split. The middle key will go up to the parent.  
[](http://www.geeksforgeeks.org/wp-content/uploads/BTreeIns6.png)

See [this](http://integrator-crimea.com/ddu0110.html) for more examples.

Following is C++ implementation of the above proactive algorithm.

// C++ program for B-Tree insertion

#include<iostream>

using namespace std;

// A BTree node

class BTreeNode

{

int \*keys; // An array of keys

int t; // Minimum degree (defines the range for number of keys)

BTreeNode \*\*C; // An array of child pointers

int n; // Current number of keys

bool leaf; // Is true when node is leaf. Otherwise false

public:

BTreeNode(int \_t, bool \_leaf); // Constructor

// A utility function to insert a new key in the subtree rooted with

// this node. The assumption is, the node must be non-full when this

// function is called

void insertNonFull(int k);

// A utility function to split the child y of this node. i is index of y in

// child array C[]. The Child y must be full when this function is called

void splitChild(int i, BTreeNode \*y);

// A function to traverse all nodes in a subtree rooted with this node

void traverse();

// A function to search a key in subtree rooted with this node.

BTreeNode \*search(int k); // returns NULL if k is not present.

// Make BTree friend of this so that we can access private members of this

// class in BTree functions

friend class BTree;

};

// A BTree

class BTree

{

BTreeNode \*root; // Pointer to root node

int t; // Minimum degree

public:

// Constructor (Initializes tree as empty)

BTree(int \_t)

{ root = NULL; t = \_t; }

// function to traverse the tree

void traverse()

{ if (root != NULL) root->traverse(); }

// function to search a key in this tree

BTreeNode\* search(int k)

{ return (root == NULL)? NULL : root->search(k); }

// The main function that inserts a new key in this B-Tree

void insert(int k);

};

// Constructor for BTreeNode class

BTreeNode::BTreeNode(int t1, bool leaf1)

{

// Copy the given minimum degree and leaf property

t = t1;

leaf = leaf1;

// Allocate memory for maximum number of possible keys

// and child pointers

keys = new int[2\*t-1];

C = new BTreeNode \*[2\*t];

// Initialize the number of keys as 0

n = 0;

}

// Function to traverse all nodes in a subtree rooted with this node

void BTreeNode::traverse()

{

// There are n keys and n+1 children, travers through n keys

// and first n children

int i;

for (i = 0; i < n; i++)

{

// If this is not leaf, then before printing key[i],

// traverse the subtree rooted with child C[i].

if (leaf == false)

C[i]->traverse();

cout << " " << keys[i];

}

// Print the subtree rooted with last child

if (leaf == false)

C[i]->traverse();

}

// Function to search key k in subtree rooted with this node

BTreeNode \*BTreeNode::search(int k)

{

// Find the first key greater than or equal to k

int i = 0;

while (i < n && k > keys[i])

i++;

// If the found key is equal to k, return this node

if (keys[i] == k)

return this;

// If key is not found here and this is a leaf node

if (leaf == true)

return NULL;

// Go to the appropriate child

return C[i]->search(k);

}

// The main function that inserts a new key in this B-Tree

void BTree::insert(int k)

{

// If tree is empty

if (root == NULL)

{

// Allocate memory for root

root = new BTreeNode(t, true);

root->keys[0] = k; // Insert key

root->n = 1; // Update number of keys in root

}

else // If tree is not empty

{

// If root is full, then tree grows in height

if (root->n == 2\*t-1)

{

// Allocate memory for new root

BTreeNode \*s = new BTreeNode(t, false);

// Make old root as child of new root

s->C[0] = root;

// Split the old root and move 1 key to the new root

s->splitChild(0, root);

// New root has two children now. Decide which of the

// two children is going to have new key

int i = 0;

if (s->keys[0] < k)

i++;

s->C[i]->insertNonFull(k);

// Change root

root = s;

}

else // If root is not full, call insertNonFull for root

root->insertNonFull(k);

}

}

// A utility function to insert a new key in this node

// The assumption is, the node must be non-full when this

// function is called

void BTreeNode::insertNonFull(int k)

{

// Initialize index as index of rightmost element

int i = n-1;

// If this is a leaf node

if (leaf == true)

{

// The following loop does two things

// a) Finds the location of new key to be inserted

// b) Moves all greater keys to one place ahead

while (i >= 0 && keys[i] > k)

{

keys[i+1] = keys[i];

i--;

}

// Insert the new key at found location

keys[i+1] = k;

n = n+1;

}

else // If this node is not leaf

{

// Find the child which is going to have the new key

while (i >= 0 && keys[i] > k)

i--;

// See if the found child is full

if (C[i+1]->n == 2\*t-1)

{

// If the child is full, then split it

splitChild(i, C[i]);

// After split, the middle key of C[i] goes up and

// C[i] is splitted into two. See which of the two

// is going to have the new key

if (keys[i] < k)

i++;

}

C[i+1]->insertNonFull(k);

}

}

// A utility function to split the child y of this node

// Note that y must be full when this function is called

void BTreeNode::splitChild(int i, BTreeNode \*y)

{

// Create a new node which is going to store (t-1) keys

// of y

BTreeNode \*z = new BTreeNode(y->t, y->leaf);

z->n = t - 1;

// Copy the last (t-1) keys of y to z

for (int j = 0; j < t-1; j++)

z->keys[j] = y->keys[j+t];

// Reduce the number of keys in y

y->n = t - 1;

// Since this node is going to have a new child,

// create space of new child

for (int j = n; j >= i+1; j--)

C[j+1] = C[j];

// Link the new child to this node

C[i+1] = z;

// A key of y will move to this node. Find location of

// new key and move all greater keys one space ahead

for (int j = n-1; j >= i; j--)

keys[j+1] = keys[j];

// Copy the middle key of y to this node

keys[i] = y->keys[t-1];

// Increment count of keys in this node

n = n + 1;

}

// Driver program to test above functions

int main()

{

BTree t(3); // A B-Tree with minium degree 3

t.insert(10);

t.insert(20);

t.insert(5);

t.insert(6);

t.insert(12);

t.insert(30);

t.insert(7);

t.insert(17);

cout << "Traversal of the constucted tree is ";

t.traverse();

int k = 6;

(t.search(k) != NULL)? cout << "\nPresent" : cout << "\nNot Present";

k = 15;

(t.search(k) != NULL)? cout << "\nPresent" : cout << "\nNot Present";

return 0;

}

Output:

Traversal of the constucted tree is 5 6 7 10 12 17 20 30

Present

Not Present

## Suffix Tree

## Prefix tree

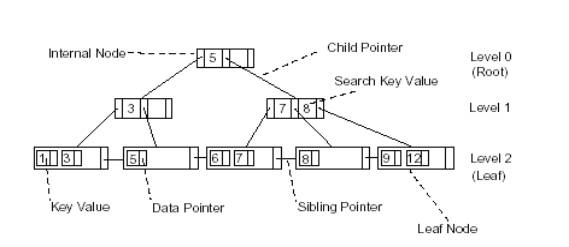
## B+ Trees

<http://www.mec.ac.in/resources/notes/notes/ds/bplus.htm>

The **B-tree** is the classic disk-based data structure for indexing records based on an ordered key set. The **B+-tree** (sometimes written B+-tree, B+tree, or just B-tree) is a variant of the original B-tree in which all records are stored in the leaves and all leaves are linked sequentially. The B+-tree is used as a (dynamic) indexing method in relational database management systems.

**B+-tree** considers all the keys in nodes except the leaves as dummies. All keys are duplicated in the leaves. This has the advantage that is all the leaves are linked together sequentially, the entire tree may be scanned without visiting the higher nodes at all.

***B+-Tree Structure***



* A B + -Tree consists of one or more blocks of data, called *node*s, linked together by pointers. The B + -Tree is a tree structure. The tree has a single node at the top, called the *root nod*e. The root node points to two or more blocks, called *child node*s. Each child nodes points to further child nodes and so on.
* The B + -Tree consists of two types of (1) *internal nodes* and (2) *leaf node*s:
  + Internal nodes point to other nodes in the tree.
  + Leaf nodes point to data in the database using *data pointer*s. Leaf nodes also contain an additional pointer, called the *sibling pointe*r, which is used to improve the efficiency of certain types of search.
* All the nodes in a B + -Tree must be at least half full except the root node which may contain a minimum of two entries. The algorithms that allow data to be inserted into and deleted from a B + -Tree guarantee that each node in the tree will be at least half full.
* Searching for a value in the B + -Tree always starts at the root node and moves downwards until it reaches a leaf node.
* Both internal and leaf nodes contain *key values* that are used to guide the search for entries in the index.
* The B + -Tree is called a *balanced tree* because every path from the root node to a leaf node is the same length. A balanced tree means that all searches for individual values require the same number of nodes to be read from the disc.

***Internal Nodes***

* An *internal node* in a B + -Tree consists of a set of *key values* and *pointer*s. The set of keys and values are ordered so that a pointer is followed by a key value. The last key value is followed by one pointer.
* Each pointer points to nodes containing values that are *less than or equal to* the value of the key immediately to its right
* The last pointer in an internal node is called the *infinity pointe*r. The infinity pointer points to a node containing key values that are greater than the last key value in the node.
* When an internal node is searched for a key value, the search begins at the leftmost key value and moves rightwards along the keys.
* If the key value is less than the sought key then the pointer to the left of the key is known to point to a node containing keys less than the sought key.
* If the key value is greater than or equal to the sought key then the pointer to the left of the key is known to point to a node containing keys between the previous key value and the current key value.

***Leaf Nodes***

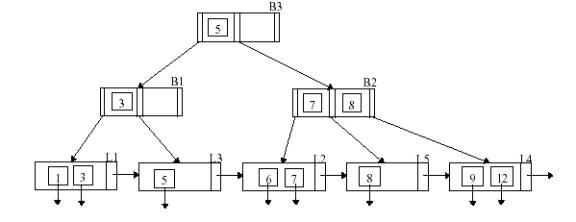
* A *leaf node* in a B + -Tree consists of a set of *key values* and *data pointer*s. Each key value has one data pointer. The key values and data pointers are ordered by the key values.
* The data pointer points to a record or block in the database that contains the record identified by the key value. For instance, in the example, above, the pointer attached to key value 7 points to the record identified by the value 7.
* Searching a leaf node for a key value begins at the leftmost value and moves rightwards until a matching key is found.
* The leaf node also has a pointer to its immediate *sibling node* in the tree. The sibling node is the node immediately to the right of the current node. Because of the order of keys in the B + -Tree the sibling pointer always points to a node that has key values that are greater than the key values in the current node.

***Order of a B + -Tree***

* The *order* of a B + -Tree is the number of keys and pointers that an internal node can contain. An order size of *m* means that an internal node can contain *m-1* keys and *m* pointers.
* The order size is important because it determines how large a B + -Tree will become.
* For example, if the order size is small then fewer keys and pointers can be placed in one node and so more nodes will be required to store the index. If the order size is large then more keys and pointers can be placed in a node and so fewer nodes are required to store the index.

**Searching a B+-Tree**

Searching a B+-Tree for a key value always starts at the root node and descends down the tree. A search for a single key value in a B+-Tree consisting of unique values will always follow one path from the root node to a leaf node.



***Searching for Key Value 6***

 · Read block *B3* from disc. ~ *read the root node*

· Is *B3* a leaf node? No ~ *its not a leaf node so the search continues*

· Is 6 <= 5? No ~ s*tep through each value in B3*

· Read block B2. ~ *when all else fails follow the infinity pointer*

· Is *B2* a leaf node? No *~ B2 is not a leaf node, continue the search*

· Is 6 <= 7? Yes *~ 6 is less than or equal to 7, follow pointer*

· Read block L2. *~ read node L2 which is pointed to by 7 in B2*

· Is *L2* a leaf node? Yes *~ L2 is a leaf node*

· Search *L2* for the key value 6. *~ if 6 is in the index it must be in L2*

***Searching for Key Value 5***

· Read block *B3* from disc. ~ *read the root node*

· Is *B3* a leaf node? No ~ *its not a leaf node so the search continues*

· Is 5 <= 5? Yes ~ s*tep through each value in B3*

· Read block *B1*. ~ *read node B1 which is pointed to by 5 in B3*

· Is *B1* a leaf node? No *~ B1 is not a leaf node, continue the search*

· Is 5 <= 3? No *~ step through each value in B1*

· Read block *L3*. *~ when all else fails follow the infinity pointer*

· Is *L3* a leaf node? Yes *~ L3 is a leaf node*

· Search *L3* for the key value 5. *~ if 5 is in the index it must be in L3*

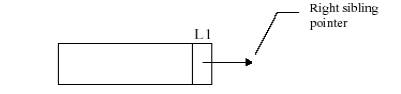
 Inserting in a B+-Tree

A B+-Tree consists of two types of node: (i) leaf nodes, which contain pointers to data records, and (ii)internal nodes, which contain pointers to other internal nodes or leaf nodes. In this example, we assume that the order size1 is 3 and that there are a maximum of two keys in each leaf node.

Insert sequence : 5, 8, 1, 7, 3, 12, 9, 6

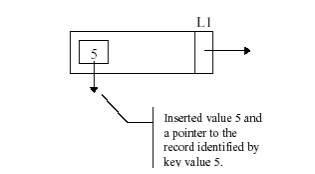
***Empty Tree***

The B+-Tree starts as a single leaf node. A leaf node consists of one or more data pointers and a pointer to its right sibling. This leaf node is empty.



***Inserting Key Value 5***

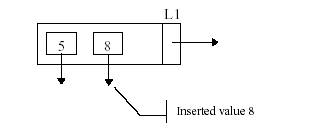
To insert a key search for the location where the key would be expected to occur. In our example the B+-Tree consists of a single leaf node, *L1*, which is empty. Hence, the key value 5 must be placed in leaf node *L1*.



***Inserting Key Value 8***

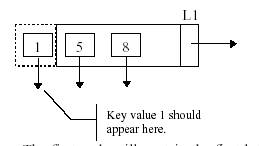
Again, search for the location where key value 8 is expected to be found. This is in leaf node *L1*.

There is room in *L1* so insert the new key.

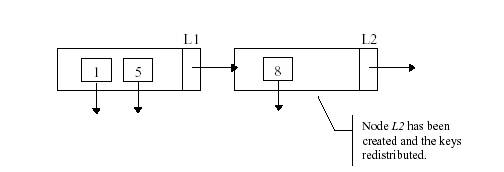
******

***Inserting Key Value 1***

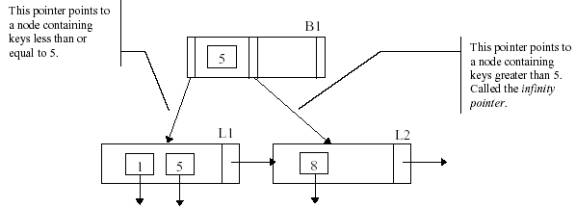
Searching for where the key value 1 should appear also results in *L1* but *L1* is now full it contains the maximum two records.



*L1* must be split into two nodes. The first node will contain the first half of the keys and the second node will contain the second half of the keys



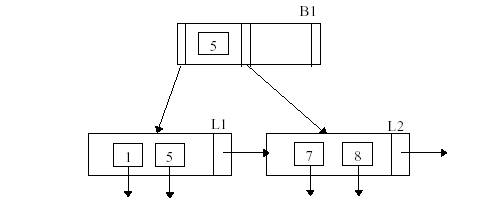
  However, we now require a new *root* node to point to each of these nodes. We create a new root node and promote the rightmost key from node *L1*.



 Each node is half full.

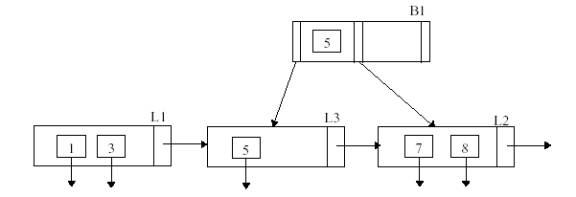
***Insert Key Value 7***

Search for the location where key 7 is expected to be located, that is, *L2*. Insert key 7 into *L2*.

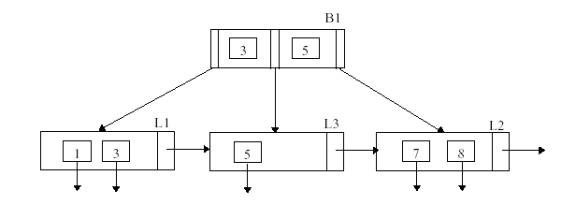


***Insert Key Value 3***

Search for the location where key 3 is expected to be found results in reading *L1*. But, *L1* is full and must be split.



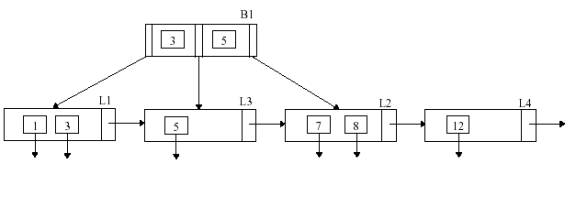
The rightmost key in *L1*, i.e. 3, must now be promoted up the tree.



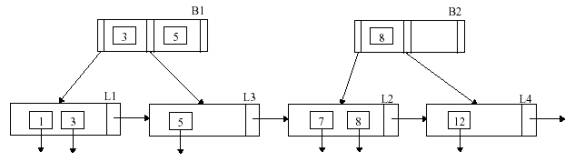
*L1* was pointed to by key 5 in *B1*. Therefore, all the key values in *B1* to the right of and including key 5 are moved to the right one place.

***Insert Key Value 12***

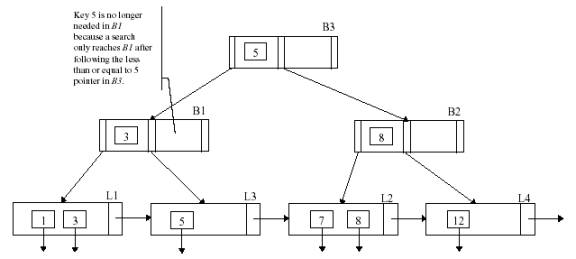
Search for the location where key 12 is expected to be found, *L2*. Try to insert 12 into *L2*. Because *L2* is full it must be split.



As before, we must promote the rightmost value of *L2* but *B1* is full and so it must be split.



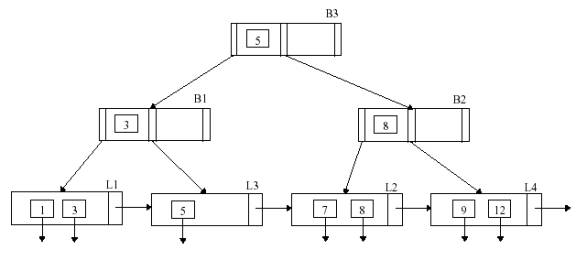
  Now the tree requires a new root node, so we promote the rightmost value of *B1* into a new node.



  The tree is still balanced, that is, all paths from the root node, *B3*, to a leaf node are of equal length.

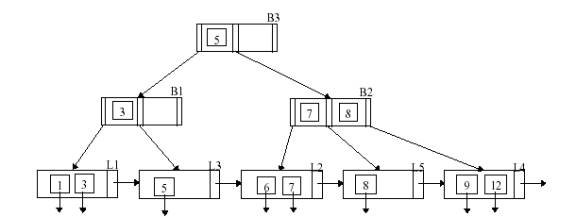
***Insert Key Value 9***

Search for the location where key value 9 would be expected to be found, *L4*. Insert key 9 into *L4*.



***Insert Key Value 6***

Key value 6 should be inserted into *L2* but it is full. Therefore, split it and promote the appropriate key value.



 Leaf block *L2* has split and the middle key, 7, has been promoted into *B2*.

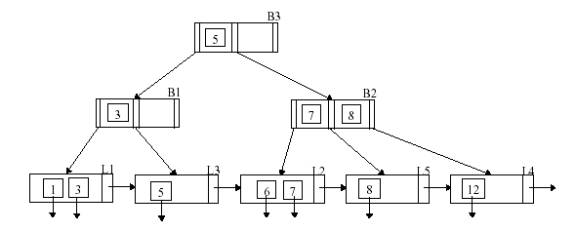
**Deleting from a B+-Tree**

Deleting entries from a B+-Tree may require some redistribution of the key values to guarantee a wellbalanced tree.

Deletion sequence: 9, 8, 12.

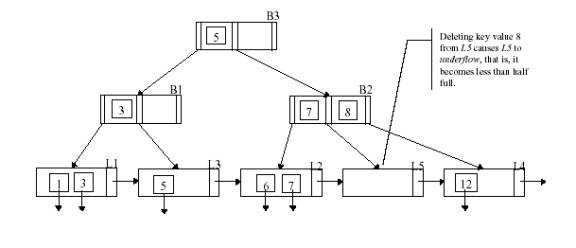
***Delete Key Value 9***

First, search for the location of key value 9, *L4*. Delete 9 from *L4*. *L4* is not less than half full and the tree is correct.

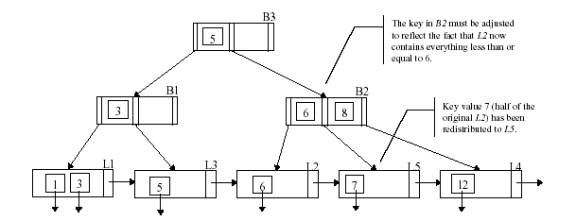


***Delete Key Value 8***

Search for key value 8, *L5*. Deleting 8 from *L5* causes *L5* to *underflow*, that is, it becomes less than half full.



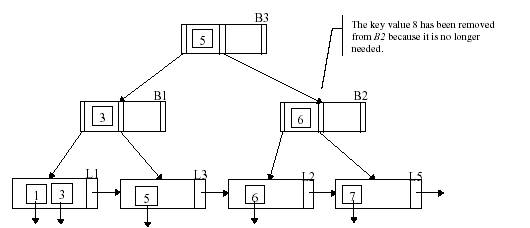
  We could remove *L5* but instead we will attempt to redistribute some of the values from *L2*. This is possible because *L2* is full and half its contents can be placed in *L5*. As some entries have been removed from *L2*, its parent *B2* must be adjusted to reflect the change.



 We can do this by removing it from the index and then adjusting the parent node *B2*.

***Deleting Key Value 12***

Deleting key value 12 from *L4* causes *L4* to underflow. However, because *L5* is already half full we cannot redistribute keys between the nodes. *L4* must be deleted from the index and *B2* adjusted to reflect the change.



  The tree is still balanced and all nodes are at least half full. However, to guarantee this property it is sometimes necessary to perform a more extensive redistribution of the data.

**Search Algorithm**

  s = Key value to be found

n = Root node

o = Order of B+-Tree

WHILE n is not a leaf node

i = 1

found = FALSE

WHILE i <= (o-1) AND NOT found

IF s <= nk[i] THEN

n = np[i]

found = TRUE

ELSE

i = i + 1

END

END

IF NOT found THEN

n = np[i]

END

END

**Insert Algorithm**

s = Key value to be inserted

Search tree for node n containing key s with path in stack p

from root(bottom) to parent of node n(top).

IF found THEN

STOP

ELSE

IF n is not full THEN

Insert s into n

ELSE

Insert s in n (\* assume n can hold s temporarily \*)

j = number of keys in n / 2

Split n to give n and n1

Put first j keys from n in n

Put remaining keys from n in n1

(k,p) = (nk[j],"pointer to n1")

REPEAT

IF p is empty THEN

Create internal node n2

Put (k,p) in n2

finished = TRUE

ELSE

n = POP p

IF n is not full THEN

Put (k,p) in n

finished = TRUE

ELSE

j = number of keys in n / 2

Split n into n and n1

Put first j keys and pointers in n into n

Put remaining keys and pointers in n into n1

(k,p) = (nk[j],"pointer to n1")

END

END

UNTIL finished

END

END

## AVL

<http://www.geeksforgeeks.org/avl-tree-set-1-insertion/>

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.

**Why AVL Trees?**  
Most of the BST operations (e.g., search, max, min, insert, delete.. etc) take O(h) time where h is the height of the BST. The cost of these operations may become O(n) for a skewed Binary tree. If we make sure that height of the tree remains O(Logn) after every insertion and deletion, then we can guarantee an upper bound of O(Logn) for all these operations. The height of an AVL tree is always O(Logn) where n is the number of nodes in the tree (See [this](http://www.youtube.com/watch?v=TbvhGcf6UJU) video lecture for proof).

**Insertion**  
To make sure that the given tree remains AVL after every insertion, we must augment the standard BST insert operation to perform some re-balancing. Following are two basic operations that can be performed to re-balance a BST without violating the BST property (keys(left) < key(root) < keys(right)).  
1) Left Rotation  
2) Right Rotation

T1, T2 and T3 are subtrees of the tree rooted with y (on left side)

or x (on right side)

y x

/ \ Right Rotation / \

x T3 – - – - – - – > T1 y

/ \ < - - - - - - - / \

T1 T2 Left Rotation T2 T3

Keys in both of the above trees follow the following order

keys(T1) < key(x) < keys(T2) < key(y) < keys(T3)

So BST property is not violated anywhere.

**Steps to follow for insertion**  
Let the newly nserted node be w  
**1)** Perform standard BST insert for w.  
**2)** Starting from w, travel up and find the first unbalanced node. Let z be the first unbalanced node, y be the child of z that comes on the path from w to z and x be the grandchild of z that comes on the path from w to z.  
**3)** Re-balance the tree by performing appropriate rotations on the subtree rooted with z. There can be 4 possible cases that needs to be handled as x, y and z can be arranged in 4 ways. Following are the possible 4 arrangements:  
a) y is left child of z and x is left child of y (Left Left Case)  
b) y is left child of z and x is right child of y (Left Right Case)  
c) y is right child of z and x is right child of y (Right Right Case)  
d) y is right child of z and x is left child of y (Right Left Case)

Following are the operations to be performed in above mentioned 4 cases. In all of the cases, we only need to re-balance the subtree rooted with z and the complete tree becomes balanced as the height of subtree (After appropriate rotations) rooted with z becomes same as it was before insertion. (See [this](http://www.youtube.com/watch?v=TbvhGcf6UJU) video lecture for proof)

**a) Left Left Case**

T1, T2, T3 and T4 are subtrees.

z y

/ \ / \

y T4 Right Rotate (z) x z

/ \ - - - - - - - - -> / \ / \

x T3 T1 T2 T3 T4

/ \

T1 T2

**b) Left Right Case**

z z x

/ \ / \ / \

y T4 Left Rotate (y) x T4 Right Rotate(z) y z

/ \ - - - - - - - - -> / \ - - - - - - - -> / \ / \

T1 x y T3 T1 T2 T3 T4

/ \ / \

T2 T3 T1 T2

**c) Right Right Case**

z y

/ \ / \

T1 y Left Rotate(z) z x

/ \ - - - - - - - -> / \ / \

T2 x T1 T2 T3 T4

/ \

T3 T4

**d) Right Left Case**

z z x

/ \ / \ / \

T1 y Right Rotate (y) T1 x Left Rotate(z) z x

/ \ - - - - - - - - -> / \ - - - - - - - -> / \ / \

x T4 T2 y T1 T2 T3 T4

/ \ / \

T2 T3 T3 T4

**C implementation**  
Following is the C implementation for AVL Tree Insertion. The following C implementation uses the recursive BST insert to insert a new node. In the recursive BST insert, after insertion, we get pointers to all ancestors one by one in bottom up manner. So we don’t need parent pointer to travel up. The recursive code itself travels up and visits all the ancestors of the newly inserted node.  
1) Perform the normal BST insertion.  
2) The current node must be one of the ancestors of the newly inserted node. Update the height of the current node.  
3) Get the balance factor (left subtree height – right subtree height) of the current node.  
4) If balance factor is greater than 1, then the current node is unbalanced and we are either in Left Left case or left Right case. To check whether it is left left case or not, compare the newly inserted key with the key in left subtree root.  
5) If balance factor is less than -1, then the current node is unbalanced and we are either in Right Right case or Right Left case. To check whether it is Right Right case or not, compare the newly inserted key with the key in right subtree root.

#include<stdio.h>

#include<stdlib.h>

// An AVL tree node

struct node

{

int key;

struct node \*left;

struct node \*right;

int height;

};

// A utility function to get maximum of two integers

int max(int a, int b);

// A utility function to get height of the tree

int height(struct node \*N)

{

if (N == NULL)

return 0;

return N->height;

}

// A utility function to get maximum of two integers

int max(int a, int b)

{

return (a > b)? a : b;

}

/\* Helper function that allocates a new node with the given key and

NULL left and right pointers. \*/

struct node\* newNode(int key)

{

struct node\* node = (struct node\*)

malloc(sizeof(struct node));

node->key = key;

node->left = NULL;

node->right = NULL;

node->height = 1; // new node is initially added at leaf

return(node);

}

// A utility function to right rotate subtree rooted with y

// See the diagram given above.

struct node \*rightRotate(struct node \*y)

{

struct node \*x = y->left;

struct node \*T2 = x->right;

// Perform rotation

x->right = y;

y->left = T2;

// Update heights

y->height = max(height(y->left), height(y->right))+1;

x->height = max(height(x->left), height(x->right))+1;

// Return new root

return x;

}

// A utility function to left rotate subtree rooted with x

// See the diagram given above.

struct node \*leftRotate(struct node \*x)

{

struct node \*y = x->right;

struct node \*T2 = y->left;

// Perform rotation

y->left = x;

x->right = T2;

// Update heights

x->height = max(height(x->left), height(x->right))+1;

y->height = max(height(y->left), height(y->right))+1;

// Return new root

return y;

}

// Get Balance factor of node N

int getBalance(struct node \*N)

{

if (N == NULL)

return 0;

return height(N->left) - height(N->right);

}

struct node\* insert(struct node\* node, int key)

{

/\* 1. Perform the normal BST rotation \*/

if (node == NULL)

return(newNode(key));

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

/\* 2. Update height of this ancestor node \*/

node->height = max(height(node->left), height(node->right)) + 1;

/\* 3. Get the balance factor of this ancestor node to check whether

this node became unbalanced \*/

int balance = getBalance(node);

// If this node becomes unbalanced, then there are 4 cases

// Left Left Case

if (balance > 1 && key < node->left->key)

return rightRotate(node);

// Right Right Case

if (balance < -1 && key > node->right->key)

return leftRotate(node);

// Left Right Case

if (balance > 1 && key > node->left->key)

{

node->left = leftRotate(node->left);

return rightRotate(node);

}

// Right Left Case

if (balance < -1 && key < node->right->key)

{

node->right = rightRotate(node->right);

return leftRotate(node);

}

/\* return the (unchanged) node pointer \*/

return node;

}

// A utility function to print preorder traversal of the tree.

// The function also prints height of every node

void preOrder(struct node \*root)

{

if(root != NULL)

{

printf("%d ", root->key);

preOrder(root->left);

preOrder(root->right);

}

}

/\* Drier program to test above function\*/

int main()

{

struct node \*root = NULL;

/\* Constructing tree given in the above figure \*/

root = insert(root, 10);

root = insert(root, 20);

root = insert(root, 30);

root = insert(root, 40);

root = insert(root, 50);

root = insert(root, 25);

/\* The constructed AVL Tree would be

30

/ \

20 40

/ \ \

10 25 50

\*/

printf("Pre order traversal of the constructed AVL tree is \n");

preOrder(root);

return 0;

}

Output:

Pre order traversal of the constructed AVL tree is

30 20 10 25 40 50

Time Complexity: The rotation operations (left and right rotate) take constant time as only few pointers are being changed there. Updating the height and getting the balance factor also take constant time. So the time complexity of AVL insert remains same as BST insert which is O(h) where h is height of the tree. Since AVL tree is balanced, the height is O(Logn). So time complexity of AVL insert is O(Logn).

The AVL tree and other self balancing search trees like Red Black are useful to get all basic operations done in O(Logn) time. The AVL trees are more balanced compared to Red Black Trees, but they may cause more rotations during insertion and deletion. So if your application involves many frequent insertions and deletions, then Red Black trees should be preferred. And if the insertions and deletions are less frequent and search is more frequent operation, then AVL tree should be preferred over Red Black Tree.

## Red Black Tree

1. A node is either red or black.
2. The root is black. (This rule is sometimes omitted. Since the root can always be changed from red to black, but not necessarily vice-versa, this rule has little effect on analysis.)
3. All leaves (NIL) are black. (All leaves are same color as the root.)
4. Both children of every red node are black.
5. Every simple path from a given node to any of its descendant leaves contains the same number of black nodes.

Contraint it gives: the path from the root to the furthest leaf is no more than twice as long as the path from the root to the nearest leaf.

## Stable Marriage Problem

Given n men and n women, where each person has ranked all members of the opposite sex with a unique number between 1 and n in order of preference, marry the men and women together such that there are no two people of opposite sex who would both rather have each other than their current partners. If there are no such people, all the marriages are "stable".

**function** stableMatching {

Initialize all *m* ∈ M and *w* ∈ W to *free*

**while** ∃ *free* man *m* who still has a woman w to propose to {

w = m's highest ranked such woman to whom he has not yet proposed

**if** w is *free*

(m, w) become *engaged*

**else** some pair (m', w) already exists

**if** w prefers m to m'

(m, w) become *engaged*

m' becomes *free*

**else**

(m', w) remain *engaged*

}

}

## Stable Roommate problem

# Memory Organization of a program

**How is memory organized?**

* Text = code
* Data = constants
* BSS(Block Started Symbol) = global and static variables
* Stack = local variables
* Heap = dynamic memory

**Data Segment=(Data + BSS + Heap)**

**Data**

The data area contains global and static variables used by the program that are ***initialized***. This segment can be further classified into **initialized read-only area** and **initialized read-write area**. For instance the string defined by char s[] = "hello world" in C and a C statement like int debug=1 outside the main would be stored in initialized read-write area. And a C statement like const char\* string = "hello world" makes the string literal "hello world" to be stored in initialized read-only area and the character pointer variable string in initialized read-write area. Ex: static int i = 10 will be stored in data segment **and** global int i = 10 will be stored in data segment.

**BSS**

The BSS segment also colloquially known as ***uninitialized*** *data* starts at the end of the data segment and contains all global variables and static variables that are **initialized to *zero* or do not have explicit initialization** in source code. For instance a variable declared static int i; would be contained in the BSS segment.

**static keyword in declaration of local variable means:**

* Available (if within scope) throughout entire program execution
* Variable is allocated from BSS, not stack
* Acts like global variable with limited scope

Operator Overloading

When to overload -> () [] operators?

When to use Operator overloading via friend function or member function?

Can virtual functions be private?

Yes it is perfectly valid to have private virtual functions. Especially used when we want subclass to be able to modify the behavior and only super class knows when the functions should be called.

Unions as Base Class

Why union can't be used in Inheritance?

The standard defines that Unions cannot be used as Base class, but is there any specific reasoning for this? As far as I understand Unions can have constructors, destructors, also member variables, and methods to operate on those varibales. In short a Union can encapsulate a datatype and state which might be accessed through member functions. Thus it in most common terms qualifies for being a class and if it can act as a class then why is it restricted from acting as a base class?

======================

Tony Park gave an answer which is pretty close to the truth. The C++ committee basically didn't think it was worth the effort to make unions a strong part of C++, similarly to the treatment of arrays as legacy stuff we had to inherit from C but didn't really want.

Unions have problems: if we allow non-POD types in unions, how do they get constructed? It can certainly be done, but not necessarily safely, and any consideration would require committee resources. And the final result would be less than satisfactory, because what is really required in a sane language is discriminated unions, and bare C unions could never be elevated to discriminated unions in way compatible with C (that I can imagine, anyhow).

To elaborate on the technical issues: since you can wrap a POD-component only union in a struct without losing anything, there's no advantage allowing unions as bases. With POD-only union components, there's no problem with explicit constructors simply assigning one of the components, nor with using a bitblit (memcpy) for compiler generated copy constructor (or assignment).

Such unions, however, aren't useful enough to bother with except to retain them so existing C code can be considered valid C++. These POD-only unions are broken in C++ because they fail to retain a vital invariant they possess in C: any data type can be used as a component type.

To make unions useful, we must allow constructable types as members. This is significant because it is not acceptable to merely assign a component in a constructor body, either of the union itself, or any enclosing struct: you cannot, for example, assign a string to an uninitialised string component.

It follows one must invent some rules for initialising union component with mem-initialisers, for example:

union X { string a; string b; X(string q) : a(q) {} };

But now the question is: what is the rule? Normally the rule is you must initialise every member and base of a class, if you do not do so explicitly, the default constructor is used for the remainder, and if one type which is not explicitly initialised does not have a default constructor, it's an error [Exception: copy constructors, the default is the member copy constructor].

Clearly this rule can't work for unions: the rule has to be instead: if the union has at least one non-POD member, you must explicitly initialise exactly one member in a constructor. In this case, no default constructor, copy constructor, assignment operator, or destructor will be generated and if any of these members are actually used, they must be explicitly supplied.

So now the question becomes: how would you write, say, a copy constructor? It is, of course quite possible to do and get right if you design your union the way, say, X-Windows event unions are designed: with the discriminant tag in each component, but you will have to use placement operator new to do it, and you will have to break the rule I wrote above which appeared at first glance to be correct!

What about default constructor? If you don't have one of those, you can't declare an uninitialised variable.

There are other cases where you can determine the component externally and use placement new to manage a union externally, but that isn't a copy constructor. The fact is, if you have N components you'd need N constructors, and C++ has a broken idea that constructors use the class name, which leaves you rather short of names and forces you to use phantom types to allow overloading to choose the right constructor .. and you can't do that for the copy constructor since its signature is fixed.

Ok, so are there alternatives? Probably, yes, but they're not so easy to dream up, and harder to convince over 100 people that it's worthwhile to think about in a three day meeting crammed with other issues.

It is a pity the committee did not implement the rule above: unions are mandatory for aligning arbitrary data and external management of the components is not really that hard to do manually, and trivial and completely safe when the code is generated by a suitable algorithm, in other words, the rule is mandatory if you want to use C++ as a compiler target language and still generate readable, portable code. Such unions with constructable members have many uses but the most important one is to represent the stack frame of a function containing nested blocks: each block has local data in a struct, and each struct is a union component, there is no need for any constructors or such, the compiler will just use placement new. The union provides alignment and size, and cast free component access. [And there is no other conforming way to get the right alignment!]

Therefore the answer to your question is: you're asking the wrong question. There's no advantage to POD-only unions being bases, and they certainly can't be derived classes because then they wouldn't be PODs. To make them useful, some time is required to understand why one should follow the principle used everywhere else in C++: missing bits aren't an error unless you try to use them.

==========

Union is a type that can be used as any one of its members depending on which member has been set - only that member can be later read.

When you derive from a type the derived type inherits the base type - the derived type can be used wherever the base type could be. If you could derive from a union the derived class could be used (not implicitly, but explicitly through naming the member) wherever any of the union members could be used, but among those members only one member could be legally accessed. The problem is the data on which member has been set is not stored in the union.

To avoid this subtle yet dangerous contradiction that in fact subverts a type system deriving from a union is not allowed.

==========

There's no technical reason why unions can't be a base class; it's just not allowed. A reasonable interpretation would be to think of the union as a struct whose members happen to potentially overlap in memory, and consider the derived class as a class that inherits from this (rather odd) struct. If you need that functionality, you can usually persuade most compilers to accept an anonymous union as a member of a struct. Here's an example, that's suitable for use as a base class. (And there's an anonymous struct in the union for good measure.)

struct V3 {

union {

struct {

float x,y,z;

};

float f[3];

};

};

The rationale for unions as a derived class is probably simpler: the result wouldn't be a union. Unions would have to be the union of all their members, and all of their bases. That's fair enough, and might open up some interesting template possibilities, but you'd have a number of limitations (all bases and members would have to be POD -- and would you be able to inherit twice, because a derived type is inherently non-POD?), this type of inheritance would be different from the other type the language sports (OK, not that this has stopped C++ before) and it's sort of redundant anyway -- the existing union functionality would do just as well.

Stroustrup says this in the D&E book:

As with void \*, programmers should know that unions ... are inherently dangerous, should be avoided wherever possible, and should be handled with special care when actually needed.

(The elision doesn't change the meaning.)

So I imagine the decision is arbitrary, and he just saw no reason to change the union functionality (it works fine as-is with the C subset of C++), and so didn't design any integration with the new C++ features. And when the wind changed, it got stuck that way.

===========

===========

Pure virtual functions may not have an inline definition. Why?

Pure virtual functions are those member functions that are virtual and have the pure-specifier ( = 0; )

Clause 10.4 paragraph 2 of C++03 tells us what an abstract class is and, as a side note, the following:

[Note: a function declaration cannot provide both a pure-specifier and a definition —end note] [Example:

struct C {

virtual void f() = 0 { }; // ill-formed

};

—end example]

For those who are not very familiar with the issue, please note that pure virtual functions can have definitions but the above-mentioned clause forbids such definitions to appear inline (lexically in-class). (For uses of defining pure virtual functions you may see, for example, this GotW)

Now for all other kinds and types of functions it is allowed to provide an in-class definition, and this restriction seems at first glance absolutely artificial and inexplicable. Come to think of it, it seems such on second and subsequent glances :) But I believe the restriction wouldn't be there if there weren't a specific reason for that. My question is: does anybody know that specific reasons?

=========

The curious =0 syntax was chosen over the obvious alternative of introducing a new keyword pure or abstract because at the time I saw no chance of getting a new keyword accepted. Had I suggested pure, Release 2.0 would have shipped without abstract classes. Given a choice between a nicer syntax and abstract classes, I chose abstract classes. Rather than risking delay and incurring the certain fights over pure, I used the tradition C and C++ convention of using 0 to represent "not there." The =0 syntax fits with my view that a function body is the initializer for a function and also with the (simplistic, but usually adequate) view of the set of virtual functions being implemented as a vector of function pointers. [ … ]

So, when choosing the syntax Bjarne was thinking of a function body as a kind of initializer part of the declarator, and =0 as an alternate form of initializer, one that indicated “no body” (or in his words, “not there”).

It stands to reason that one cannot both indicate “not there” and have a body – in that conceptual picture.

Or, still in that conceptual picture, having two initializers.

=========

=========

How are array and pointer types handled internally in C compilers? ( int \*a; vs. int a[]; )

From the comp.lang.C FAQ:

... whenever an array appears in an expression, the compiler implicitly generates a pointer to the array's first element, just as if the programmer had written &a[0]. (The exceptions are when the array is the operand of a sizeof or & operator, or is a string literal initializer for a character array...)

... Given an array a and pointer p, an expression of the form a[i] causes the array to decay into a pointer, following the rule above, and then to be subscripted just as would be a pointer variable in the expression p[i] (although the eventual memory accesses will be different ...

Given declarations of

char a[] = "hello";

char \*p = "world";

... when the compiler sees the expression a[3], it emits code to start at the location a, move three past it, and fetch the character there. When it sees the expression p[3], it emits code to start at the location p, fetch the pointer value there, add three to the pointer, and finally fetch the character pointed to. In other words, a[3] is three places past (the start of) the object named a, while p[3] is three places past the object pointed to by p.

==========

I agree with sepp2k's answer and Mark Rushakoff's comp.lang.c FAQ quote. Let me add some important differences between the two declarations and a common trap.

1. When you define a as an array (in a context other than a function's argument, which is a special case) you can't write a = 0; or a++; because a is not an lvalue (a value that can appear on the left of an assignment operator).
2. The array definition reserves space, whereas the pointer doesn't. Therefore, sizeof(array) will return the memory space needed for storing all the array's elements (for instance 10 times four bytes for an array of 10 integers on a 32-bit architecture), whereas sizeof(pointer) will only return the memory space required for storing that pointer (for instance 8 bytes in a 64-bit architecture).
3. When you prepend pointer or append array declarations things definitely diverge. For instance, int \*\*a is a pointer to a pointer to an integer. It can be used as a two-dimensional array (with rows of varying sizes) by allocating an array of pointers to the rows and making each one point to memory for storing integers. To access a[2][3] the compiler will fetch the pointer in a[2] and then move three elements past the location it points to in order to access the value. Contrast this with b[10][20] which is an array of 10 elements, each of which is an array of 20 integers. To access b[2][3] the compiler will offset the beginning of the array's memory area by multiplying 2 by the size of 20 integers and adding the size of 3 more integers.

===========

a and b are both arrays of ints. a[0] is not a memory location containing a memory address, it is a memory location containing an int.

Arrays and pointers are neither identical nor interchangeable. Arrays are equivalent to pointers iff when an lvalue of type array-of-T which appears in an expression decays (with three exceptions) into a pointer to its first element; the type of the resultant pointer is pointer-to-T. This becomes clear when looking at the assembly output for related code. The three exceptions, fyi, are when the array is an operand of sizeof or & or a literal string initializer for a character array.

If you would picture this:

char a[] = "hello";

char \*p = "world";

would result in data structures which could be represented like this:

+---+---+---+---+---+---+

a: | h | e | l | l | o |\0 |

+---+---+---+---+---+---+

+-----+ +---+---+---+---+---+---+

p: | \*======> | w | o | r | l | d |\0 |

+-----+ +---+---+---+---+---+---+

and realize that a reference like x[3] produces different code depending on whether x is a pointer or an array. a[3] for the compiler means: start at the location a and move three past it and fetch the char there. p[3] means go to the location p, dereference the value there, move three past it and fetch the char there.

=======

========

Where are static variables stored (in C/C++)?

Where your statics go depends on if they are 0 initialized or not. 0 initialized static data goes in .BSS (Block Started by Symbol), non 0 initialized data goes in .DATA

Data declared in a compilation unit will go into the .BSS or the .Data of that files output. Initialised data in BSS, uninitalised in DATA.

The difference between static and global data comes in the inclusion of symbol information in the file. Compilers tend to include the symbol information but only mark the global information as such.

The linker respects this information. The symbol information for the static variables is either discarded or mangled so that static variables can still be referenced in some way (with debug or symbol options). In neither case can the compilation units gets affected as the linker resolves local references first.

===

In fact, a variable is tuple (storage, scope, type, address, value):

storage : where is it stored, for example data, stack, heap...

scope : who can see us, for example global, local...

type : what is our type, for example int, int\*...

address : where are we located

value : what is our value

Local scope could mean local to either the translational unit (source file), the function or the block depending on where its defined. To make variable visible to more than one function, it definitely has to be in either DATA or the BSS area (depending on whether its initialized explicitly or not, respectively). Its then scoped accordingly to either all function(s) or function(s) within source file.

=======

======

What are uses of the C++ construct “placement new”?

It is also used for embedded programming, where IO devices are often mapped to specific memory addresses

It allows you to do your own memory management. Usually this will get you at best marginally improved performance, but sometimes it's a big win

I've used it when constructing objects in a shared memory segment.

Its usefull when building your own container like objects.

a. you want to create objects in memory shared between two different processes

b. you want objects to be created in non-pageable memory

c. you want to seperate memory allocation from construction eg. in implementing a std::vector<> (see std::vector<>::reserve)

this is the kind of problem where you could use the flyweight design pattern quite effectively

http://www.glenmccl.com/nd\_cmp.htm

=====

=====

Why are global anonymous unions required to be declared as static?

C++ 0x draft

9.5.6 Anonymous unions declared in a named namespace or in the global namespace shall be declared static.

Why?

Update-

the best explanation so far might be this:

If the same global anonymous union is encountered in two translation units (say, via a header file), then how can the One Definition Rule be satisfied? Are the two definitions treated as the same and merged together? Or are the two definitions treated as different? If they are treated as the same, then the compiler is presumably doing 'magic' it doesn't otherwise do for other entities. If they are treated as the same, then the compiler is doing so without the explicit consent of the programmer... so I suppose explicit consent is being forced by requiring it to be declared as static.

============

===========

What are POD types in C++?

POD stands for Plain Old Data - that is, a struct (or class) with no members except data members. Wikipedia goes into a bit more detail and defines a POD in C++ as "A Plain Old Data Structure in C++ is an aggregate class that contains only PODS as members, has no user-defined destructor, no user-defined copy assignment operator, and no nonstatic members of pointer-to-member type."

=====

A POD is a type (including classes) where the C++ compiler guarantees that there will be no "magic" going on in the structure: for example hidden pointers to vtables, offsets that get applied to the address when it is cast to other types (at least if the target's POD too), constructors, or destructors. Roughly speaking, a type is a POD when the only things in it are built-in types and combinations of them. The result is something that "acts like" a C type.

Less informally:

\* int, char, wchar\_t, bool, float, double are PODs, as are long/short and signed/unsigned versions of them.

\* pointers (including pointer-to-function and pointer-to-member) are PODs,

\* enums are PODs

\* a const or volatile POD is a POD.

\* a class, struct or union of PODs is a POD provided that all members are public, and it has no base class and no constructors, destructors, or virtual methods. Static members don't stop something being a POD under this rule.

\* Wikipedia is wrong to say that a POD cannot have members of type pointer-to-member. Or rather, it's correct for the C++98 wording, but TC1 made explicit that pointers-to-member are POD.

Here's what the C++ standard says:

3.9(10): "Arithmetic types (3.9.1), enumeration types, pointer types, and pointer to member types (3.9.2) and cv-qualified versions of these types (3.9.3) are collectively caller scalar types. Scalar types, POD-struct types, POD-union types (clause 9), arrays of such types and cv-qualified versions of these types (3.9.3) are collectively called POD types"

9(4): "A POD-struct is an aggregate class that has no non-static data members of type non-POD-struct, non-POD-union (or array of such types) or reference, and has no user-define copy operator and no user-defined destructor. Similarly a POD-union is an aggregate union that has no non-static data members of type non-POD-struct, non-POD-union (or array of such types) or reference, and has no user-define copy operator and no user-defined destructor.

8.5.1(1): "An aggregate is an array or class (clause 9) with no user-declared constructors (12.1), no private or protected non-static data members (clause 11), no base classes (clause 10) and no virtual functions (10.3)."

===========

With C++, Plain Old Data doesn't just mean that things like int, char, etc are the only types used. Plain Old Data really means in practice that you can take a struct memcpy it from one location in memory to another and things will work exactly like you would expect (i.e. not blow up). This breaks if your class, or any class your class contains, has as a member that is a pointer or a reference or a class that has a virtual function. Essentially, if pointers have to be involved somewhere, its not Plain Old Data.

==========

=========

One Definition Rule?

http://en.wikipedia.org/wiki/One\_Definition\_Rule

In short the ODR states that:

1. In any translation unit, a template, type, function, or object can have no more than one definition. Some of these can have any number of declarations. A definition provides an instance.

2. In the entire program, an object or non-inline function cannot have more than one definition; if an object or function is used, it must have exactly one definition. You can declare an object or function that is never used, in which case you don't have to provide a definition. In no event can there be more than one definition.

3. Some things, like types, templates, and extern inline functions, can be defined in more than one translation unit. For a given entity, each definition must be the same. Non-extern objects and functions in different translation units are different entities, even if their names and types are the same.

==========

========

Determining the Size of a Class Object

http://www.cprogramming.com/tutorial/size\_of\_class\_object.html

There are many factors that decide the size of an object of a class in C++. These factors are:

1. Size of all non-static data members

2. Order of data members

3. Byte alignment or byte padding

4. Size of its immediate base class

5. The existence of virtual function(s) (Dynamic polymorphism using virtual functions).

6. Compiler being used

7. Mode of inheritance (virtual inheritance)

STL

-containers- sequence, associative,

-algorithms- initialization, sorting, searching, transforming the contents.

-iterators – Random, Bidirectional, Forward, Input, Output

-Other stl elements – allocators (allocator class), predicates, comparison functions, & function objects.

-predicates- unary, binary.

Comparison- Comp class

# General Questions answered

**Q. Why looking for a change?**

To be honest, I am looking at my career growth in two terms; **technology vertical** and **monetory**. I want my career path to be going **true** **north**.

I learned a lot about introducing new products on my job, and after 6 years of experience in my current position, I'm looking for an company where I can contribute more on working with **challenging tasks**, **enhance my technical skills & experience**, and hence have a true north career growth.

**<Your company>** seemed like an excellent match for my skills and experience that I have and not able to fully utilize them in my present job.

**Q. Tell me about yourself?**

Currently I am working with Symantec in Security Technology and Response Group. I am working in CDT team. CDT stands for ContainerDeliveryTeam. CDT provides 3 major components that are used by both the Consumer and Enterprise teams in Symantec.

Decomposer, Typer and DecABI.

**Decomposer** is a collection of libraries having a well defined API for recursive processing of container types like ZIP, RAR,7z etc. Decomposer engines provide cross platform solutions for Symantec applications and dependent component technologies to access encoded/compressed data, and data within containers.

**Typer** provides a fast and easy managed Content Type Identification for file types. It internally uses DataSource and foundation libraries that are also owned by our team.

**DecABI** is a Binary interface over Decomposer to provide a single binary(dll) to our clients. We have also added the wide char support in DecABI which Decomposer lacks.

Some of the main features of Decomposer are:

1. Rich file type support. To be exact Decomposer has 22 engine, with which you can process more than 50 file types.
2. In memory decomposition, via our implemented In-memory file system.
3. Rich API support for writing client FS, Callbacks for the client to get control at critical points.
4. Multithreading support.
5. Unicode support

Before joining Symantec I completed my Masters from IIT Bombay. There I got the chance to work with lot of professors under many projects. Worth mentioning will be my MTP, that was in PBX systems and SIP optimization for wireless mediums. My seminar in Service Oriented Architecture. Lot of other projects like Postgres reverse indexing, WRED simulations.

I have completed by BE from DAVV, Indore, One of the most reputed colleges in MP. There also I was lucky to work on good projects like Grid Computing.

About my personal fields of interest are Computer Networks, Mobile networks, Designing and working on research problems.

Grid computing is a computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers to model a virtual computer architecture that is able to distribute process execution across a parallel infrastructure. Grids provide the ability to perform computations on large data sets, by breaking them down into many smaller ones, or provide the ability to perform many more computations at once than would be possible on a single computer, by modeling a parallel division of labor between processes.

We used the Globus Toolkit to create a grid of computers on LAN. The Globus toolkit is open source toolkit that provides functionalities to create robust, reliable, scalable and secure grids. We used GAF4J (Grid Application Framework for Java) to create state-full weather services. We also used Cogkit to develop clinets for the grid services.

In pervasive computing environment the applications must become more context-aware, flexible, and autonomous. To facilitate the programming of such applications, infrastructure is required to gather, store, and disseminate context information to applications.

In this seminar we focused on three areas of making applications context aware: The **physical layer** of the pervasive computing, where sensors that are used to collect the raw context information. We present the **layered, and object oriented model** used to represent the context information. We also presented the **middleware architecture** for context-aware applications and its functioning in pervasive computing.

Design and analysis of Algorithms, Data mining, Database Management Systems, Computer Networks, Mobile Computing, Principles and practice of Distributed Computing, Qos in Networks, Applied Economics.

1. If I want to buy something like a book or a tool, how does the process work (how hard is it?). What's the cost limit before the approval must go up the management chain?
2. What's the noise level like during the day?
3. How many meetings am I expected to attend, and how long do they usually last?
4. Is there a dress code?
5. Can I work from home sometimes?
6. Does it matter when I work, as long as I come to meetings?
7. How many projects have succeeded/failed in the last five years? To what do you attribute the failures?

# Computer Algorithms

Chapter 1

1. Selection sort
2. nCm
3. Towers of Hanoi
4. Permutation Generator
5. n!
6. Fibonacci series
   1. Find nth element
   2. Print series upto nth element
7. Q10, 11, 12
8. BigO, Omega, Theta Little o, omega
9. Magic Square
10. Exponentiate

Chapter 2

1. Stack
   1. Array implementation
   2. Linked list implementation
2. Queue
   1. Circular queue
      1. Problem of checking full and empty
   2. When storing n-1 elements
      1. FULL( (r==n-1 && f==0) || (f-r==1))
      2. Empty(f==r)
3. Linked list- merge, reverse, find circle
4. Double-ended queue
5. Tree: A tree is a finite set of one or more nodes s.t. there is a specially designated node called root, and the remaining nodes are partitioned into n>=0 disjoint sets T1, T2,…Tn, where each of these sets is a tree.
   1. Terms: Degree of a node, Height/depth of tree, forest
6. Binary Trees
   1. Tree of degree 2
   2. The maximum number of nodes on level i of a binary tree is 2(i-1). Also the maximum number of nodes in a binary tee of depth k is 2k-1. k >0.
   3. In a complete binary tree on n nodes, there are (n+1)/2 leaves and (n-1)/2 internal nodes
7. Sequential representation of Tree
   1. Root is numbered 0, and rest nodes from left to right sequentially.
   2. parnent(i) is at (i-1)/2 if i !=0. For i ==0 is root.
   3. lchild(i) is at 2i+1 if 2i+1<=n. If 2i+1 > n, then i has no left child.
   4. rchild(i) is at 2i+2, if 2i+2<=n. If 2i+2>n, then i has no right child.
8. Binary search trees
   1. Searching a BST
   2. Finding the kth smallest element: leftSize=1 + number of nodes in the left subtree.
   3. Insertion into a BST
   4. Deletion from BST
   5. Balanced Trees: AVL trees, 2-3, Red Black, B-trees, Splay tree
   6. Q5,
   7. Merge two BST

If A has no children, just insert the node normally into B.

If the root of A < the root of B:

A1 = A.right

A.right = NULL

Merge A with B.left

Merge A1 with B

Otherwise:

A1 = A.left

A.left = NULL

Merge A1 with B

Merge A with B.right

* 1. Determine is a binary tree is a BST

1. Priority queues
   1. Heap: A max heap is an **almost compete** binary tree with the property that the value at each node is at least as large as the values at its children.
   2. Heap: Array implementation
      1. Insert O(logn)
      2. Delete O(logn)
      3. Adjust O(logn)
      4. Heapify O(n): Given an array of n elements we can convert it into a heap in O(n) time.
      5. Min-max heap, Deap, Leftist tree, Binomial heap, Fibonacci heap, 2-3 tree, Red-Black tree.
2. Sets and Disjoint set union
   1. G
3. Graphs

C++ FAQ

Chapter 8: 1, 3, 6

Chapter 9:

Chapter 10: 2, 8, 9

# Windows API

## Synchronization Functions

The following functions are used in synchronization.

* [Asynchronous functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#asynchronous_functions)
* [Condition variable and SRW lock functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#condition_variable_and_SRW_lock_functions)
* [**Critical section functions**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#critical_section_functions)
* [**Event functions**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#event_functions)
* [One-time initialization functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#one-time_initialization_functions)
* [Interlocked Functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#interlocked_functions)
* [**Mutex functions**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#mutex_functions)
* [Private namespace functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#private_namespace_functions)
* [**Semaphore functions**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#semaphore_functions)
* [Singly-linked list functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#singly-linked_list_functions)
* [Synchronization barrier functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#synchronization_barrier_functions)
* [Timer-queue timer functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#timer-queue_timer_functions)
* [Wait functions](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#wait_functions)
* [**Waitable-timer functions**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686360%28v=vs.85%29.aspx#waitable_timer_functions)

### Asynchronous functions

|  |  |
| --- | --- |
| **Asynchronous function** | **Description** |
| [**APCProc**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681947%28v=vs.85%29.aspx) | An application-defined callback function used with the [**QueueUserAPC**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684954%28v=vs.85%29.aspx) function. |
| [**GetOverlappedResult**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683209%28v=vs.85%29.aspx) | Retrieves the results of an overlapped operation. |
| [**GetOverlappedResultEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh448542%28v=vs.85%29.aspx) | Retrieves the results of an overlapped operation within a specified timeout interval. |
| [**QueueUserAPC**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684954%28v=vs.85%29.aspx) | Adds a user-mode asynchronous procedure call (APC) object to the APC queue of the specified thread. |

### Condition variable and SRW lock functions

|  |  |
| --- | --- |
| **Condition variable and SRW lock function** | **Description** |
| [**AcquireSRWLockExclusive**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681930%28v=vs.85%29.aspx) | Acquires a slim reader/writer (SRW) lock in exclusive mode. |
| [**AcquireSRWLockShared**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681934%28v=vs.85%29.aspx) | Acquires a slim reader/writer (SRW) lock in shared mode. |
| [**InitializeConditionVariable**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683469%28v=vs.85%29.aspx) | Initializes a condition variable. |
| [**InitializeSRWLock**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683483%28v=vs.85%29.aspx) | Initialize a slim reader/writer (SRW) lock. |
| [**ReleaseSRWLockExclusive**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685076%28v=vs.85%29.aspx) | Releases a slim reader/writer (SRW) lock that was acquired in exclusive mode. |
| [**ReleaseSRWLockShared**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685080%28v=vs.85%29.aspx) | Releases a slim reader/writer (SRW) lock that was acquired in shared mode. |
| [**SleepConditionVariableCS**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686301%28v=vs.85%29.aspx) | Sleeps on the specified condition variable and releases the specified critical section as an atomic operation. |
| [**SleepConditionVariableSRW**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686304%28v=vs.85%29.aspx) | Sleeps on the specified condition variable and releases the specified lock as an atomic operation. |
| [**TryAcquireSRWLockExclusive**](http://msdn.microsoft.com/en-us/library/windows/desktop/dd405523%28v=vs.85%29.aspx) | Attempts to acquire a slim reader/writer (SRW) lock in exclusive mode. If the call is successful, the calling thread takes ownership of the lock. |
| [**TryAcquireSRWLockShared**](http://msdn.microsoft.com/en-us/library/windows/desktop/dd405524%28v=vs.85%29.aspx) | Attempts to acquire a slim reader/writer (SRW) lock in shared mode. If the call is successful, the calling thread takes ownership of the lock. |
| [**WakeAllConditionVariable**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687076%28v=vs.85%29.aspx) | Wake all threads waiting on the specified condition variable. |
| [**WakeConditionVariable**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687080%28v=vs.85%29.aspx) | Wake a single thread waiting on the specified condition variable. |

### Critical section functions

|  |  |
| --- | --- |
| **Critical section function** | **Description** |
| [**DeleteCriticalSection**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682552%28v=vs.85%29.aspx) | Releases all resources used by an unowned critical section object. |
| [**EnterCriticalSection**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682608%28v=vs.85%29.aspx) | Waits for ownership of the specified critical section object. |
| [**InitializeCriticalSection**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683472%28v=vs.85%29.aspx) | Initializes a critical section object. |
| [**InitializeCriticalSectionAndSpinCount**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683476%28v=vs.85%29.aspx) | Initializes a critical section object and sets the spin count for the critical section. |
| [**InitializeCriticalSectionEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683477%28v=vs.85%29.aspx) | Initializes a critical section object with a spin count and optional flags. |
| [**LeaveCriticalSection**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684169%28v=vs.85%29.aspx) | Releases ownership of the specified critical section object. |
| [**SetCriticalSectionSpinCount**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686197%28v=vs.85%29.aspx) | Sets the spin count for the specified critical section. |
| [**TryEnterCriticalSection**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686857%28v=vs.85%29.aspx) | Attempts to enter a critical section without blocking. |

### Event functions

|  |  |
| --- | --- |
| **Event function** | **Description** |
| [**CreateEvent**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682396%28v=vs.85%29.aspx) | Creates or opens a named or unnamed event object. |
| [**CreateEventEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682400%28v=vs.85%29.aspx) | Creates or opens a named or unnamed event object and returns a handle to the object. |
| [**OpenEvent**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684305%28v=vs.85%29.aspx) | Opens an existing named event object. |
| [**PulseEvent**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684914%28v=vs.85%29.aspx) | Sets the specified event object to the signaled state and then resets it to the nonsignaled state after releasing the appropriate number of waiting threads. |
| [**ResetEvent**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685081%28v=vs.85%29.aspx) | Sets the specified event object to the nonsignaled state. |
| [**SetEvent**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686211%28v=vs.85%29.aspx) | Sets the specified event object to the signaled state. |

### One-time initialization functions

|  |  |
| --- | --- |
| **One-time initialization function** | **Description** |
| [**InitOnceBeginInitialize**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683487%28v=vs.85%29.aspx) | Begins one-time initialization. |
| [**InitOnceComplete**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683491%28v=vs.85%29.aspx) | Completes one-time initialization. |
| [**InitOnceExecuteOnce**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683493%28v=vs.85%29.aspx) | Executes the specified function successfully one time. No other threads that specify the same one-time initialization structure can execute this function while it is being executed by the current thread. |
| [**InitOnceInitialize**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683495%28v=vs.85%29.aspx) | Initializes a one-time initialization structure. |

### Interlocked Functions

|  |  |
| --- | --- |
| **Interlocked function** | **Description** |
| [**InterlockedAdd**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683504%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONG** values. |
| [**InterlockedAddAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683507%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedAddRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683513%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedAddNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972629%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedAdd64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683506%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONGLONG** values. |
| [**InterlockedAddAcquire64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683510%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONGLONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedAddRelease64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683514%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONGLONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedAddNoFence64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972630%28v=vs.85%29.aspx) | Performs an atomic addition operation on the specified **LONGLONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedAnd**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683516%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONG** values. |
| [**InterlockedAndAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683539%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedAndRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683543%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedAndNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972634%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedAnd8**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683532%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **char** values. |
| [**InterlockedAnd8Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683535%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **char** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedAnd8Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683537%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **char** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedAnd8NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972633%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **char** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedAnd16**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683518%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **SHORT** values. |
| [**InterlockedAnd16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683521%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **SHORT** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedAnd16Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683525%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **SHORT** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedAnd16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972631%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **SHORT** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedAnd64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683527%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONGLONG** values. |
| [**InterlockedAnd64Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683529%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONGLONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedAnd64Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683530%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONGLONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedAnd64NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972632%28v=vs.85%29.aspx) | Performs an atomic AND operation on the specified **LONGLONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedBitTestAndComplement**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh802759%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and complements it. |
| [**InterlockedBitTestAndComplement64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972635%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG64** value and complements it. The operation is atomic |
| [**InterlockedBitTestAndResetAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972636%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and sets it to 0. The operation is atomic, and it is performed with acquire memory ordering semantics |
| [**InterlockedBitTestAndResetRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972637%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and sets it to 0. The operation is atomic, and it is performed using memory release semantics |
| [**InterlockedBitTestAndSetAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972638%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and sets it to 1. The operation is atomic, and it is performed with acquire memory ordering semantics |
| [**InterlockedBitTestAndSetRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972639%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and sets it to 1. The operation is atomic, and it is performed with release memory ordering semantics |
| [**InterlockedBitTestAndReset**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683546%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and sets it to 0. |
| [**InterlockedBitTestAndReset64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683547%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG64** value and sets it to 0. |
| [**InterlockedBitTestAndSet**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683549%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG** value and sets it to 1. |
| [**InterlockedBitTestAndSet64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683549%28v=vs.85%29.aspx) | Tests the specified bit of the specified **LONG64** value and sets it to 1. |
| [**InterlockedCompare64Exchange128**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683553%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares the specified 64-bit values and exchanges with the specified 128-bit value based on the outcome of the comparison. |
| [**InterlockedCompare64ExchangeAcquire128**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683557%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares the specified 64-bit values and exchanges with the specified 128-bit value based on the outcome of the comparison. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedCompare64ExchangeRelease128**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683558%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares the specified 64-bit values and exchanges with the specified 128-bit value based on the outcome of the comparison. The operation is performed with release memory ordering semantics. |
| [**InterlockedCompareExchange**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683560%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 32-bit values and exchanges with another 32-bit value based on the outcome of the comparison. |
| [**InterlockedCompareExchangeAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683564%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 32-bit values and exchanges with another 32-bit value based on the outcome of the comparison. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedCompareExchangeRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683574%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 32-bit values and exchanges with another 32-bit value based on the outcome of the comparison. The exchange is performed with release memory ordering semantics. |
| [**InterlockedCompareExchangeNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972645%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 32-bit values and exchanges with another 32-bit value based on the outcome of the comparison. The operation is performed atomically, but without using memory barriers |
| [**InterlockedCompareExchange64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683562%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 64-bit values and exchanges with another 64-bit value based on the outcome of the comparison. |
| [**InterlockedCompareExchangeAcquire64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683566%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 64-bit values and exchanges with another 64-bit value based on the outcome of the comparison. The exchange is performed with acquire memory ordering semantics. |
| [**InterlockedCompareExchangeRelease64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683576%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 64-bit values and exchanges with another 64-bit value based on the outcome of the comparison. The exchange is performed with release memory ordering semantics. |
| [**InterlockedCompareExchangeNoFence64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972646%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 64-bit values and exchanges with another 64-bit value based on the outcome of the comparison. The operation is performed atomically, but without using memory barriers |
| [**InterlockedCompareExchange16**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972641%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 16-bit values and exchanges with another 16-bit value based on the outcome of the comparison |
| [**InterlockedCompareExchange16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972642%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 16-bit values and exchanges with another 16-bit value based on the outcome of the comparison. The operation is performed with acquire memory ordering semantics |
| [**InterlockedCompareExchange16Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972644%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 16-bit values and exchanges with another 16-bit value based on the outcome of the comparison. The exchange is performed with release memory ordering semantics |
| [**InterlockedCompareExchange16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972643%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 16-bit values and exchanges with another 16-bit value based on the outcome of the comparison. The operation is performed atomically, but without using memory barriers |
| [**InterlockedCompareExchange128**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972640%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified 128-bit values and exchanges with another 128-bit value based on the outcome of the comparison |
| [**InterlockedCompareExchangePointer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683568%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified pointer values. The function compares two specified pointer values and exchanges with another pointer value based on the outcome of the comparison. |
| [**InterlockedCompareExchangePointerAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683570%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified pointer values. The function compares two specified pointer values and exchanges with another pointer value based on the outcome of the comparison. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedCompareExchangePointerRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683571%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified pointer values. The function compares two specified pointer values and exchanges with another pointer value based on the outcome of the comparison. The operation is performed with release memory ordering semantics. |
| [**InterlockedCompareExchangePointerNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972647%28v=vs.85%29.aspx) | Performs an atomic compare-and-exchange operation on the specified values. The function compares two specified pointer values and exchanges with another pointer value based on the outcome of the comparison. The operation is performed atomically, but without using memory barriers |
| [**InterlockedDecrement**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683580%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 32-bit variable as an atomic operation. |
| [**InterlockedDecrementAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683583%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 32-bit variable as an atomic operation. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedDecrementRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683586%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 32-bit variable as an atomic operation. The operation is performed with release memory ordering semantics. |
| [**InterlockedDecrementNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972652%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 32-bit variable as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedDecrement16**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972648%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 16-bit variable as an atomic operation |
| [**InterlockedDecrement16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972649%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 16-bit variable as an atomic operation. The operation is performed with acquire memory ordering semantics |
| [**InterlockedDecrement16Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972651%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 16-bit variable as an atomic operation. The operation is performed with release memory ordering semantics |
| [**InterlockedDecrement16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972650%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 16-bit variable as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedDecrement64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683581%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 64-bit variable as an atomic operation. |
| [**InterlockedDecrementAcquire64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683585%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 64-bit variable as an atomic operation. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedDecrementRelease64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683588%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 64-bit variable as an atomic operation. The operation is performed with release memory ordering semantics. |
| [**InterlockedDecrementNoFence64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972653%28v=vs.85%29.aspx) | Decrements (decreases by one) the value of the specified 64-bit variable as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedExchange**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683590%28v=vs.85%29.aspx) | Sets a 32-bit variable to the specified value as an atomic operation. |
| [**InterlockedExchangeAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683594%28v=vs.85%29.aspx) | Sets a 32-bit variable to the specified value as an atomic operation. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedExchangeNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972659%28v=vs.85%29.aspx) | Sets a 64-bit variable to the specified value as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedExchange8**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972656%28v=vs.85%29.aspx) | Sets an 8-bit variable to the specified value as an atomic operation |
| [**InterlockedExchange16**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh802760%28v=vs.85%29.aspx) | Sets a 16-bit variable to the specified value as an atomic operation. |
| [**InterlockedExchange16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972654%28v=vs.85%29.aspx) | Sets a 16-bit variable to the specified value as an atomic operation. The operation is performed using acquire memory ordering semantics |
| [**InterlockedExchange16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972655%28v=vs.85%29.aspx) | Sets a 16-bit variable to the specified value as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedExchange64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683593%28v=vs.85%29.aspx) | Sets a 64-bit variable to the specified value as an atomic operation. |
| [**InterlockedExchangeAcquire64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683596%28v=vs.85%29.aspx) | Sets a 32-bit variable to the specified value as an atomic operation. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedExchangeNoFence64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972660%28v=vs.85%29.aspx) | Sets a 64-bit variable to the specified value as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedExchangePointer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683609%28v=vs.85%29.aspx) | Atomically exchanges a pair of pointer values. |
| [**InterlockedExchangePointerAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683611%28v=vs.85%29.aspx) | Atomically exchanges a pair of pointer values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedExchangePointerNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972661%28v=vs.85%29.aspx) | Atomically exchanges a pair of addresses. The operation is performed atomically, but without using memory barriers |
| [**InterlockedExchangeSubtract**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh871477%28v=vs.85%29.aspx) | Performs an atomic subtraction of two values. |
| [**InterlockedExchangeAdd**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683597%28v=vs.85%29.aspx) | Performs an atomic addition of two 32-bit values. |
| [**InterlockedExchangeAddAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683601%28v=vs.85%29.aspx) | Performs an atomic addition of two 32-bit values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedExchangeAddRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683605%28v=vs.85%29.aspx) | Performs an atomic addition of two 32-bit values. The operation is performed with release memory ordering semantics. |
| [**InterlockedExchangeAddNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972657%28v=vs.85%29.aspx) | Performs an atomic addition of two 32-bit values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedExchangeAdd64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683599%28v=vs.85%29.aspx) | Performs an atomic addition of two 64-bit values. |
| [**InterlockedExchangeAddAcquire64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683604%28v=vs.85%29.aspx) | Performs an atomic addition of two 64-bit values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedExchangeAddRelease64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683607%28v=vs.85%29.aspx) | Performs an atomic addition of two 64-bit values. The operation is performed with release memory ordering semantics. |
| [**InterlockedExchangeAddNoFence64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972658%28v=vs.85%29.aspx) | Performs an atomic addition of two 64-bit values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedIncrement**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683614%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 32-bit variable as an atomic operation. |
| [**InterlockedIncrementAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683618%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 32-bit variable as an atomic operation. The operation is performed using acquire memory ordering semantics. |
| [**InterlockedIncrementRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683622%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 32-bit variable as an atomic operation. The operation is performed using release memory ordering semantics. |
| [**InterlockedIncrementNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972667%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 32-bit variable as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedIncrement16**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972662%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 16-bit variable as an atomic operation |
| [**InterlockedIncrement16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972663%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 16-bit variable as an atomic operation. The operation is performed using acquire memory ordering semantics |
| [**InterlockedIncrement16Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972665%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 16-bit variable as an atomic operation. The operation is performed using release memory ordering semantics |
| [**InterlockedIncrement16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972664%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 16-bit variable as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedIncrement64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683615%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 64-bit variable as an atomic operation. |
| [**InterlockedIncrementAcquire64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683620%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 64-bit variable as an atomic operation. The operation is performed using acquire memory ordering semantics. |
| [**InterlockedIncrementRelease64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683624%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 64-bit variable as an atomic operation. The operation is performed using release memory ordering semantics. |
| [**InterlockedIncrementNoFence64**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972668%28v=vs.85%29.aspx) | Increments (increases by one) the value of the specified 64-bit variable as an atomic operation. The operation is performed atomically, but without using memory barriers |
| [**InterlockedOr**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683626%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONG** values. |
| [**InterlockedOrAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683643%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedOrRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683646%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedOrNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972672%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedOr8**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683637%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **char** values. |
| [**InterlockedOr8Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683639%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **char** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedOr8Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683640%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **char** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedOr8NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972671%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **char** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedOr16**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683627%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **SHORT** values. |
| [**InterlockedOr16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683629%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **SHORT** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedOr16Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683631%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **SHORT** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedOr16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972669%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **SHORT** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedOr64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683633%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONGLONG** values. |
| [**InterlockedOr64Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683634%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONGLONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedOr64Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683636%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONGLONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedOr64NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972670%28v=vs.85%29.aspx) | Performs an atomic OR operation on the specified **LONGLONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedXor**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684021%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONG** values. |
| [**InterlockedXorAcquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684117%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedXorRelease**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684118%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedXorNoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972677%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONG** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedXor8**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684110%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **char** values. |
| [**InterlockedXor8Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684112%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **char** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedXor8Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684113%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **char** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedXor8NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972676%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **char** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedXor16**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684024%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **SHORT** values. |
| [**InterlockedXor16Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684026%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **SHORT** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedXor16Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684033%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **SHORT** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedXor16NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972674%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **SHORT** values. The operation is performed atomically, but without using memory barriers |
| [**InterlockedXor64**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684104%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONGLONG** values. |
| [**InterlockedXor64Acquire**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684107%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONGLONG** values. The operation is performed with acquire memory ordering semantics. |
| [**InterlockedXor64Release**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684108%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONGLONG** values. The operation is performed with release memory ordering semantics. |
| [**InterlockedXor64NoFence**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972675%28v=vs.85%29.aspx) | Performs an atomic XOR operation on the specified **LONGLONG** values. The operation is performed atomically, but without using memory barriers |

### Mutex functions

|  |  |
| --- | --- |
| **Mutex function** | **Description** |
| [**CreateMutex**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682411%28v=vs.85%29.aspx) | Creates or opens a named or unnamed mutex object. |
| [**CreateMutexEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682418%28v=vs.85%29.aspx) | Creates or opens a named or unnamed mutex object and returns a handle to the object. |
| [**OpenMutex**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684315%28v=vs.85%29.aspx) | Opens an existing named mutex object. |
| [**ReleaseMutex**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685066%28v=vs.85%29.aspx) | Releases ownership of the specified mutex object. |

### Private namespace functions

|  |  |
| --- | --- |
| **Private namespace function** | **Description** |
| [**AddSIDToBoundaryDescriptor**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681937%28v=vs.85%29.aspx) | Adds a new security identifier (SID) to the specified boundary descriptor. |
| [**AddIntegrityLabelToBoundaryDescriptor**](http://msdn.microsoft.com/en-us/library/windows/desktop/ee175818%28v=vs.85%29.aspx) | Adds a new required security identifier (SID) to the specified boundary descriptor. |
| [**ClosePrivateNamespace**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682026%28v=vs.85%29.aspx) | Closes an open namespace handle. |
| [**CreateBoundaryDescriptor**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682121%28v=vs.85%29.aspx) | Creates a boundary descriptor. |
| [**CreatePrivateNamespace**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682419%28v=vs.85%29.aspx) | Creates a private namespace. |
| [**DeleteBoundaryDescriptor**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682549%28v=vs.85%29.aspx) | Deletes the specified boundary descriptor. |
| [**OpenPrivateNamespace**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684318%28v=vs.85%29.aspx) | Opens a private namespace. |

### Semaphore functions

|  |  |
| --- | --- |
| **Semaphore function** | **Description** |
| [**CreateSemaphore**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682438%28v=vs.85%29.aspx) | Creates or opens a named or unnamed semaphore object. |
| [**CreateSemaphoreEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682446%28v=vs.85%29.aspx) | Creates or opens a named or unnamed semaphore object and returns a handle to the object. |
| [**OpenSemaphore**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684326%28v=vs.85%29.aspx) | Opens an existing named semaphore object. |
| [**ReleaseSemaphore**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685071%28v=vs.85%29.aspx) | Increases the count of the specified semaphore object by a specified amount. |

### Singly-linked list functions

|  |  |
| --- | --- |
| **Singly-linked list function** | **Description** |
| [**InitializeSListHead**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683482%28v=vs.85%29.aspx) | Initializes the head of a singly linked list. |
| [**InterlockedFlushSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683612%28v=vs.85%29.aspx) | Flushes the entire list of items in a singly linked list. |
| [**InterlockedPopEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683648%28v=vs.85%29.aspx) | Removes an item from the front of a singly linked list. |
| [**InterlockedPushEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684020%28v=vs.85%29.aspx) | Inserts an item at the front of a singly linked list. |
| [**InterlockedPushListSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh448545%28v=vs.85%29.aspx) | Inserts a singly-linked list at the front of another singly linked list. |
| [**InterlockedPushListSListEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh972673%28v=vs.85%29.aspx) | Inserts a singly-linked list at the front of another singly linked list. Access to the lists is synchronized on a multiprocessor system. This version of the method does not use the [\_\_fastcall](http://msdn.microsoft.com/en-us/library/windows/desktop/6xa169sk%28v=vs.85%29.aspx) calling convention |
| [**QueryDepthSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684916%28v=vs.85%29.aspx) | Retrieves the number of entries in the specified singly linked list. |
| [**RtlFirstEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa904931%28v=vs.85%29.aspx) | Retrieves the first entry in a singly linked list. |
| [**RtlInitializeSListHead**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa904932%28v=vs.85%29.aspx) | Initializes the head of a singly linked list. Applications should call [**InitializeSListHead**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683482%28v=vs.85%29.aspx) instead. |
| [**RtlInterlockedFlushSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa904933%28v=vs.85%29.aspx) | Flushes the entire list of items in a singly linked list. Applications should call [**InterlockedFlushSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683612%28v=vs.85%29.aspx) instead. |
| [**RtlInterlockedPopEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa904934%28v=vs.85%29.aspx) | Removes an item from the front of a singly linked list. Applications should call [**InterlockedPopEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms683648%28v=vs.85%29.aspx) instead. |
| [**RtlInterlockedPushEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa904935%28v=vs.85%29.aspx) | Inserts an item at the front of a singly linked list. Applications should call [**InterlockedPushEntrySList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684020%28v=vs.85%29.aspx) instead. |
| [**RtlQueryDepthSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa904936%28v=vs.85%29.aspx) | Retrieves the number of entries in the specified singly linked list. Applications should call [**QueryDepthSList**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684916%28v=vs.85%29.aspx) instead. |

### Synchronization barrier functions

|  |  |
| --- | --- |
| **Synchronization barrier function** | **Description** |
| [**DeleteSynchronizationBarrier**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh706887%28v=vs.85%29.aspx) | Deletes a synchronization barrier. |
| [**EnterSynchronizationBarrier**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh706889%28v=vs.85%29.aspx) | Enters a synchronization barrier and waits for the appropriate number of threads to rendezvous at the barrier. |
| [**InitializeSynchronizationBarrier**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh706890%28v=vs.85%29.aspx) | Initializes a new synchronization barrier. |

### Timer-queue timer functions

|  |  |
| --- | --- |
| **Timer-queue timer function** | **Description** |
| [**ChangeTimerQueueTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682004%28v=vs.85%29.aspx) | Updates a timer-queue timer. |
| [**CreateTimerQueue**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682483%28v=vs.85%29.aspx) | Creates a queue for timers. |
| [**CreateTimerQueueTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682485%28v=vs.85%29.aspx) | Creates a timer-queue timer. |
| [**DeleteTimerQueue**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682565%28v=vs.85%29.aspx) | Deletes a timer queue. |
| [**DeleteTimerQueueEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682568%28v=vs.85%29.aspx) | Deletes a timer queue. |
| [**DeleteTimerQueueTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682569%28v=vs.85%29.aspx) | Cancels a timer-queue timer. |

### Wait functions

|  |  |
| --- | --- |
| **Wait function** | **Description** |
| [**MsgWaitForMultipleObjects**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684242%28v=vs.85%29.aspx) | Waits until one or all of the specified objects are in the signaled state or the time-out interval elapses. The objects can include input event objects. |
| [**MsgWaitForMultipleObjectsEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684245%28v=vs.85%29.aspx) | Waits until one or all of the specified objects are in the signaled state, an I/O completion routine or asynchronous procedure call (APC) is queued to the thread, or the time-out interval elapses. The array of objects can include input event objects. |
| [**RegisterWaitForSingleObject**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms685061%28v=vs.85%29.aspx) | Directs a wait thread in the thread pool to wait on the object. |
| [**SignalObjectAndWait**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686293%28v=vs.85%29.aspx) | Signals one object and waits on another object as a single operation. |
| [**UnregisterWait**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686870%28v=vs.85%29.aspx) | Cancels a registered wait operation. |
| [**UnregisterWaitEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686876%28v=vs.85%29.aspx) | Cancels a registered wait operation. |
| [**WaitForMultipleObjects**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687025%28v=vs.85%29.aspx) | Waits until one or all of the specified objects are in the signaled state or the time-out interval elapses. |
| [**WaitForMultipleObjectsEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687028%28v=vs.85%29.aspx) | Waits until one or all of the specified objects are in the signaled state, an I/O completion routine or asynchronous procedure call (APC) is queued to the thread, or the time-out interval elapses. |
| [**WaitForSingleObject**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687032%28v=vs.85%29.aspx) | Waits until the specified object is in the signaled state or the time-out interval elapses. |
| [**WaitForSingleObjectEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687036%28v=vs.85%29.aspx) | Waits until the specified object is in the signaled state, an I/O completion routine or asynchronous procedure call (APC) is queued to the thread, or the time-out interval elapses. |
| [**WaitOnAddress**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh706898%28v=vs.85%29.aspx) | Waits for the value at the specified address to change. |
| [**WaitOrTimerCallback**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687066%28v=vs.85%29.aspx) | An application-defined function that serves as the starting address for a timer callback or a registered wait callback. |
| [**WakeByAddressAll**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh706899%28v=vs.85%29.aspx) | Wakes all threads waiting for the value of an address to change. |
| [**WakeByAddressSingle**](http://msdn.microsoft.com/en-us/library/windows/desktop/hh706900%28v=vs.85%29.aspx) | Wakes a thread waiting for the value of an address to change. |

### Waitable-timer functions

|  |  |
| --- | --- |
| **Waitable-timer function** | **Description** |
| [**CancelWaitableTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681985%28v=vs.85%29.aspx) | Sets the specified waitable timer to the inactive state. |
| [**CreateWaitableTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682492%28v=vs.85%29.aspx) | Creates or opens a waitable timer object. |
| [**CreateWaitableTimerEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms682494%28v=vs.85%29.aspx) | Creates or opens a waitable timer object and returns a handle to the object. |
| [**OpenWaitableTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684337%28v=vs.85%29.aspx) | Opens an existing named waitable timer object. |
| [**SetWaitableTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686289%28v=vs.85%29.aspx) | Activates the specified waitable timer. |
| [**SetWaitableTimerEx**](http://msdn.microsoft.com/en-us/library/windows/desktop/dd405521%28v=vs.85%29.aspx) | Activates the specified waitable timer and provides context information for the timer. . |
| [**TimerAPCProc**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686786%28v=vs.85%29.aspx) | Application-defined timer completion routine used with the [**SetWaitableTimer**](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686289%28v=vs.85%29.aspx) function. |

## Pipes

The following function is used with anonymous pipes.

|  |  |
| --- | --- |
| **Function** | **Description** |
| [**CreatePipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365152%28v=vs.85%29.aspx) | Creates an anonymous pipe. |

The following functions are used with named pipes.

|  |  |
| --- | --- |
| **Function** | **Description** |
| [**CallNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365144%28v=vs.85%29.aspx) | Connects to a message-type pipe, writes to and reads from the pipe, and then closes the pipe. |
| [**ConnectNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365146%28v=vs.85%29.aspx) | Enables a named pipe server process to wait for a client process to connect to an instance of a named pipe. |
| [**CreateNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365150%28v=vs.85%29.aspx) | Creates an instance of a named pipe and returns a handle for subsequent pipe operations. A client process connects to a named pipe by using the [**CreateFile**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa363858%28v=vs.85%29.aspx) or [**CallNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365144%28v=vs.85%29.aspx) function. |
| [**DisconnectNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365166%28v=vs.85%29.aspx) | Disconnects the server end of a named pipe instance from a client process. |
| [**GetNamedPipeClientComputerName**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365437%28v=vs.85%29.aspx) | Retrieves the client computer name for the specified named pipe. |
| [**GetNamedPipeClientProcessId**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365440%28v=vs.85%29.aspx) | Retrieves the client process identifier for the specified named pipe. |
| [**GetNamedPipeClientSessionId**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365442%28v=vs.85%29.aspx) | Retrieves the client session identifier for the specified named pipe. |
| [**GetNamedPipeHandleState**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365443%28v=vs.85%29.aspx) | Retrieves information about a specified named pipe. |
| [**GetNamedPipeInfo**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365445%28v=vs.85%29.aspx) | Retrieves information about the specified named pipe. |
| [**GetNamedPipeServerProcessId**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365446%28v=vs.85%29.aspx) | Retrieves the server process identifier for the specified named pipe. |
| [**GetNamedPipeServerSessionId**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365569%28v=vs.85%29.aspx) | Retrieves the server session identifier for the specified named pipe. |
| [**ImpersonateNamedPipeClient**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa378618%28v=vs.85%29.aspx) | Impersonates a named-pipe client application. |
| [**PeekNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365779%28v=vs.85%29.aspx) | Copies data from a named or anonymous pipe into a buffer without removing it from the pipe. |
| [**SetNamedPipeHandleState**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365787%28v=vs.85%29.aspx) | Sets the read mode and the blocking mode of the specified named pipe. |
| [**TransactNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365790%28v=vs.85%29.aspx) | Combines the functions that write a message to and read a message from the specified named pipe into a single network operation. |
| [**WaitNamedPipe**](http://msdn.microsoft.com/en-us/library/windows/desktop/aa365800%28v=vs.85%29.aspx) | Waits until either a time-out interval elapses or an instance of the specified named pipe is available for a connection. |