Given a undirected graph, clone it. Now if the undirected graph has the neighbors with the nodes as same data - how do you make sure you create the exact same branches and also how do you make sure you don't run into loops for the exact node. He gave a empty directed graph and asked me write code after that.

from programcreek.com

class UndirectedGraphNode {

int label;

ArrayList<UndirectedGraphNode> neighbors;

UndirectedGraphNode(int x) {

label = x;

neighbors = new ArrayList<UndirectedGraphNode>();

}

}

public class Solution {

public UndirectedGraphNode cloneGraph(UndirectedGraphNode node) {

if(node == null)

return null;

LinkedList<UndirectedGraphNode> queue = new LinkedList<UndirectedGraphNode>();

HashMap<UndirectedGraphNode, UndirectedGraphNode> map =

new HashMap<UndirectedGraphNode, UndirectedGraphNode>();

UndirectedGraphNode newHead = new UndirectedGraphNode(node.label);

queue.add(node); //first clone node

map.put(node, newnode);

while(!queue.isEmpty()) {

UndirectedGraphNode current = queue.pop();

ArrayList<UndirectedGraphNode> currentNeighbors = current.neighbors;

for(UndirectedGraphNode a : currentNeighbors) {

if(!map.containsKey(a)) {

UndirectedGraphNode copy = new UndirectedGraphNode(a.label);

map.put(a, copy);

map.get(current).neighbors.add(copy);

queue.add(a);

}

else {

map.get(current).neighbors.add(map.get(a));

}

}

}

return newhead;

}

}

Given a N different open and close braces in a string "( { [ } ] )". How do you check whether the string has matching braces.

1 Maintain a stack of characters.

2 Whenever you find opening braces '(', '{' OR '[' push it on the stack.

3 Whenever you find closing braces ')', '}' OR ']' , check if top of stack is corresponding opening

bracket, if yes, then pop the stack, else break the loop and return false.

4 Repeat steps 2 - 3 until end of the string.

public boolean CheckBraces(String str) {

if(str == null || str.length() == 0)

return false;

char[] charArray = str.toCharArray();

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

if(char[i] == ‘() || char[i] == ‘{‘ || char[i] == ‘[‘)

theStack.push(char[i]);

else {

char previous = theStack.pop();

if(!isMatch(previous, char[i])

return false;

}

}

return true;

}

public boolean isMatch(char first, char last) {

if(first == ‘(‘ && last == ‘)’)

return true;

if(first == ‘[‘ && last == ‘]’)

return true;

if(first == ‘{‘ && last == ‘}’)

return true;

else

return false;

}

Given a unsorted array. Create a balanced BTREE

Print a tree like (Parent ( leftchild (leftchild, rightchild), rightchild(leftchild,rightchild) )

Pre-order traversal

private void preOrder(Node localRoot) {

if(localroot!=null) {

System.out.print(“ “ + localRoot.data);

preOrder(localRoot.leftChild);

preOrder(localRoot.rightChild);

}

}

Design Short URL

Given a List with duplicate Strings, how do you remove duplicate Strings

Both for reserving order or not

import java.util.\*;

public class RemoveDupStr {

public static void main(String[] args) {

ArrayList<String> duplicateList = new ArrayList<String>();

duplicateList.add(“Android”);

duplicateList.add(“Android”);

duplicateList.add(“iOS”);

duplicateList.add(“Windows mobile”);

System.out.println(duplicateList);

//HashSet<String> listToSet = new HashSet<String>(duplicateList);

LinkedHashSet<String> listToSet = new LinkedHashSet<String>(duplicateList);

ArrayList<String> nonDupList = new ArrayList<String>(listToSet);

System.out.priintln(nonDupList);

}

}

How do you design a Maze and what kind of data structures you use for Maze. Now print the shortest path from start to end point.

if it’s a rectangular grid and most grid squares will contain something, a good data structure is a 2D array.

However, if the maze is not a rectangle, or if a majority of the cells in a large maze don’t actually contain any useful info, a good data structure is a graph.

The 2D array’s benefit is that it’s very easy to draw and fairly straight-forward to process.

The graph structure benefit is that it takes up a lot less space. It’s the only structure which can represent random geometry and the processing is fairly straightforward.

In case of “unweighed Graph” use BFS() to return the shortest path from start to end point.

Algorithm and design a ranking system

Design a class where you can add elements, and return the mean of the latest N elements.

How to protect data when you transfer it online. e.g. You need to submit your username and password from your client to the server and there is people who is listening to the information.

Build a pseudorandom maze. Write a method to deep copy a graph. Given an n x n matrix of numbers, how do I find a number?

Merge sorted, non-overlapping list of intervals with another interval [(1,3), (5,10), (12,30)] + (9,31) = [(1,3), (5,31)]”

Given [ (1,3), (3,5), (7,9), (12,30), (32,42) ] and (9,31)

Starting with 9. Find a set with overlap otherwise append to solution.

Check 9 < 3 No. Sol = [ (1,3) ]

Check 9 < 5 No. Sol = [ (1,3), (3,5) ]

Check 9 < 9 No. Sol = [ (1,3), (3,5), (7,9) ]

Check 9 < 30 Yes.

------> Check 9 < 12 Yes. Sol = [ (1,3), (3,5), (7,9), (9,????) ]

Check 31 > 30 Yes. Sol = [ (1,3), (3,5), (7,9), (9,????) ], right=31

Check 31 < 32 Yes. Sol = [ (1,3), (3,5), (7,9), (9,31), (32,42) ]

Design a cache with O(1) search time and delete time

Graph traversal, Dijkstra

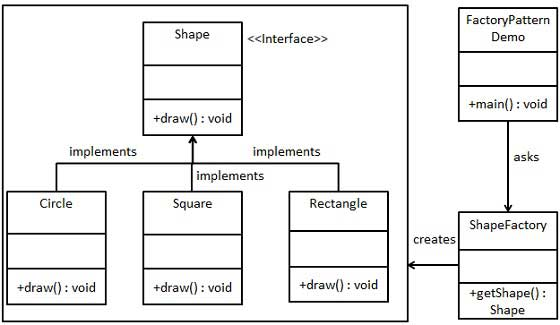
Design the classes to draw different shapes

From tutorials point.com

Factory Pattern

Factory pattern is one of most used design pattern in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.



Step 1

Create an interface.

Shape.java

public interface Shape {

void draw();

}

Step 2

Create concrete classes implementing the same interface.

Rectangle.java

public class Rectangle implements Shape {

@Override

public void draw() {

System.out.println("Inside Rectangle::draw() method.");

}

}

Square.java

public class Square implements Shape {

@Override

public void draw() {

System.out.println("Inside Square::draw() method.");

}

}

Circle.java

public class Circle implements Shape {

@Override

public void draw() {

System.out.println("Inside Circle::draw() method.");

}

}

Step 3

Create a Factory to generate object of concrete class based on given information.

ShapeFactory.java

public class ShapeFactory {

//use getShape method to get object of type shape

public Shape getShape(String shapeType){

if(shapeType == null){

return null;

}

if(shapeType.equalsIgnoreCase("CIRCLE")){

return new Circle();

} else if(shapeType.equalsIgnoreCase("RECTANGLE")){

return new Rectangle();

} else if(shapeType.equalsIgnoreCase("SQUARE")){

return new Square();

}

return null;

}

}

Step 4

Use the Factory to get object of concrete class by passing an information such as type.

FactoryPatternDemo.java

public class FactoryPatternDemo {

public static void main(String[] args) {

ShapeFactory shapeFactory = new ShapeFactory();

//get an object of Circle and call its draw method.

Shape shape1 = shapeFactory.getShape("CIRCLE");

//call draw method of Circle

shape1.draw();

//get an object of Rectangle and call its draw method.

Shape shape2 = shapeFactory.getShape("RECTANGLE");

//call draw method of Rectangle

shape2.draw();

//get an object of Square and call its draw method.

Shape shape3 = shapeFactory.getShape("SQUARE");

//call draw method of circle

shape3.draw();

}

}

First question was in-order traversal of BST. I wrote the code. Then asked me two check if two different BSTs have same inorder traversal. I suggested to store first inorder traversal into vector, and while traversing second BST check if the element values match.

Second interviewer gave me array, with elements first strictly increasing, then strictly decreasing. Asked me to find the largest number. I suggested to use binary search. O(logN) time complexity, with no additional space required. After implementing the code he asked another question. Given infinite doubly linked list, and separate list of elements, which are linked to some linked list elements. Find out from the list of elements how many neighboring groups are formed. I suggested to use Hashtable, to make look-up process quick, and traverse the elements and check if they are neighboring elements. O(N) time and space complexity. Implemented the code, and time ran out to test if with different cases.

convert tree to html

how to design a algorithm which could run in a large computer systems?

Designed a method to identify utf-8 characters.

you got a fence, you need to paint the boards with black and white, but can not have 3 or more boards same color in a roll. how many ways do you have.

Given two sorted arrays with N elements each, find the median of their union in O(log n)

From geeksforgeeks

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

Example:

ar1[] = {1, 12, 15, 26, 38}

ar2[] = {2, 13, 17, 30, 45}

For above two arrays m1 = 15 and m2 = 17

For the above ar1[] and ar2[], m1 is smaller than m2. So median is present in one of the following two subarrays.

[15, 26, 38] and [2, 13, 17]

Let us repeat the process for above two subarrays:

m1 = 26 m2 = 13.

m1 is greater than m2. So the subarrays become

[15, 26] and [13, 17]

Now size is 2, so median = (max(ar1[0], ar2[0]) + min(ar1[1], ar2[1]))/2

= (max(15, 13) + min(26, 17))/2

= (15 + 17)/2

= 16

Implementation:

#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 (i == 0 || ar2[j] > ar1[i-1])

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);

}

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

int main()

{

int ar1[] = {1, 12, 15, 26, 38};

int ar2[] = {2, 13, 17, 30, 45};

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;

}

Time Complexity: O(logn)

Algorithmic Paradigm: Divide and Conquer

(1)I have a linked-list and a node has a digit.

For example, 1->2->4->6->8.

1 is a head and 8 is a tail.

It means the number 12,468.

I want to add 1.

Then, it should be 1->2->4->6->9.

Implement it.

what is the time complexity.

(2) explain abstract and interface.

what are they different?

give me an example.

(3) Bill has a fifty billion dollar.

you have a dollar.

your money become 2 times larger than a year before. (ex: 1->2->4->8->...)

How long year does it take that yours is more than Bill's.

give me an answer without calculator.

1. Recurse to the last element and return the carry over. Add carry over to the current element and return carry over. If the calling function receives a carryover as 1, then add a new node to the linked list at the start of the list and set it to 1.

2.

3. 50,000,000,000 = 50\*10^9 = 50 \* 10^3 \* 10^3 \* 10^3 = 50\* 2^30 = 2^5.5 \* 2^30 = 2^35.5. Assuming that you know that 1000 is 2^10, and roughly guessing that 50 = 2^5.5, you can answer it as 35.5 years.

How to delete two rows from the table in database ?

Note: Delete only first two rows from the Database.

<table style="width:300px" border="2px">

<tr>

<th>USERNAME</th>

<th>LOCATION</th>

</tr>

<tr>

<td>zac</td>

<td>california</td>

</tr>

<tr>

<td>zac</td>

<td>california</td>

</tr>

<tr>

<td>zac</td>

<td>california</td>

</tr>

<tr>

<td>zac</td>

<td>california</td>

</tr>

</table>

delete from table\_name limit 2;

Give a string of words. Find anagram words and put them into the same list, then output the list of anagram words lists.

input: [add, dad, care, race]

output: [[add, dad], [care, race]]

Describe and code and efficient algorithm to decide if two rectangles in a coordinate plane are overlapping or not. Include data structure definition and any assumptions you make about the coordinate plane (location of the origin etc).