08/05/2018

Took time to write the following override functions.

All the three functions mentioned below are public functions.

* hashCode method:
  + C is upper case in the method name. hash**C**ode
  + hashCode method return an **integer.**
  + hashCode **has no input arguments.**

Example:

public int hashCode(){  
 int sum = 0;  
 for(char ch:this.orgStr.toCharArray()){  
 sum = sum + ch;  
 }  
 return sum;  
}

* equals method:
  + equals is all lower case.
  + equals method returns a **Boolean.**
  + Equals method accepts an **Object as argument.**
  + Need to check for **instanceof** method.

Example

@Override  
public boolean equals(Object obj){  
 if(obj == this)  
 return true;  
 if(!(obj instanceof Node))  
 return false;  
 int compare= orgStr.compareTo(((Node) obj).orgStr);  
 return (compare==0?true:false);  
}

* compareTo
  + compareTo is camel cased with **T** in upper in compare**T**o.
  + compareTo requires that the class **implements Comparable<Type>.**
  + compareTo returns a int.
  + compareTo accepts the type defined in the Comparable generics of the class.

@Override  
public int compareTo(Node o){  
 return sortedStr.compareTo(((Node) o).sortedStr);  
}

* ArraysList.toArray( new <Type>[ArrayList.size()) function:
  + Is **safe**. It returns a new copy of original list with no reference to original list.
  + It returns an Object array which need to be casted to original list type.

Example

* + ArrayList.toArray would table an new array of size ArrayList.

08/07/2018

binarySearch:

* For binary Search the condition should be left == right
* Check for boundary conditions, least value and highest values should return -1
* Make sure the upper search is from mid+1 since mid is already searched for.

LinearSearchForKthOrderStatistic

* The partitioned function returns only the rank of random pivot with in the subset of elements. To return on
* Need to get the indexOf the pivot as the recursion should happen based on the indexOf partitioned element.
* IndexOfPartitioned = left + rankOfPartitioned -1;
* Make sure to add left when generating random number

BFS and DFS:

* Be careful not to visit already visited nodes.
* For BFS clear a class to keep track of previous element to get path to the node.

TwoMedianSortedArray

* Three Base Cases
  + Case 1: First array ran out of elements. Return the k th order statistic (indx1 +k-1]
  + Case 2: Second array ran out of elements. Return the kth order statistic [indx2+k-1]
  + Case 3: k=1, by reducing number of elements in each recursion we arrived at the 1 st order statistic from indx1 and indx2. Return the smaller of the two.
* Edge case to consider.
  + Case when one array is smaller than the other, We do not need to check as long as the smaller array becomes equal to larger array.
  + Case of even and odd should only be handled in the first call, as we are looking for 3rd order and 4th order statistics.