TT

2/7/22

1.Inversion length 3

private static int count(int[] arr) {  
 int cnt = 0;  
 int n = arr.length;  
 for (int i= 0; i < n; ++i) {  
 int small = 0;  
 for (int j = i+1; j < n; ++j) {  
 if (arr[j] < arr[i]) ++small;  
 }  
 int great = 0;  
 for (int k = i - 1; k >= 0; --k) {  
 if (arr[k] > arr[i]) ++great;  
 }  
 cnt += small \* great;  
 }  
 return cnt;  
}

2.FindShortestLexicographicallySubstringWith K’s 1 (s only contains ‘0’, ‘1’)

private static String findShortest(String s, int k) {  
 if (s == null || s.length() < k)  
 return null;  
 List<Integer> index = new ArrayList<>();  
 for (int i = 0; i < s.length(); ++i) {  
 if (s.charAt(i) == '1')  
 index.add(i);  
 }  
 if (index.size() < k) return null;  
 String cur = "";  
 String res = s;  
 for (int i = 0; i <= index.size() - k; ++i) {  
 cur = s.substring(index.get(i), index.get(i + k - 1) + 1);  
 if (cur.length() < res.length()) {  
 res = cur;  
 } else if (cur.length() == res.length() && cur.compareTo(res) < 0) {  
 res = cur;  
 }  
 }  
 return res;  
}

3.Ancestor Name

private static String[] sort(String[] names) {  
 List<String[]> list = new ArrayList<>();  
 for (String name : names) {  
 int idx = name.lastIndexOf(' ');  
 String giveName = name.substring(0, idx);  
 String romanNum = name.substring(idx + 1);  
 int val = *romanToInteger*(romanNum);  
 list.add(new String[]{giveName, Integer.*toString*(val), name});  
 }  
 Collections.*sort*(list, (a, b) -> a[0].equals(b[0]) ? Integer.*compare*(Integer.*valueOf*(a[1]), Integer.*valueOf*(b[1])) : a[0].compareTo(b[0]));  
 String[] res = new String[names.length];  
 for (int i = 0; i < list.size(); ++i) {  
 res[i] = list.get(i)[2];  
 }  
 return res;  
}  
  
private static int romanToInteger(String roman) {  
 Map<Character, Integer> map = new HashMap<>();  
 map.put('I', 1);  
 map.put('V', 5);  
 map.put('X', 10);  
 map.put('L', 50);  
 map.put('C', 100);  
 map.put('D', 500);  
 map.put('M', 1000);  
 char[] c = roman.toCharArray();  
 int res = 0, i = 0;  
 for (; i < c.length - 1; ++i) {  
 if (map.get(c[i]) >= map.get(c[i+1]))  
 res += map.get(c[i]);  
 else  
 res -= map.get(c[i]);  
 }  
 res += map.get(c[i]);  
 return res;  
}

4.Dominos 3D

private static int numTilings3(int n) {  
 long[] dp = new long[n+2];  
 long[] dpa = new long[n + 2];  
 dp[0] = 1;  
 dp[1] = 2;  
 dpa[0] = 0;  
 dpa[1] = 4;  
 for (int i = 2; i <= n; ++i) {  
 dpa[i] = dpa[i - 1] + 4 \* dp[i - 1] % *mod*;  
 dp[i] = (dpa[i - 1] + 2 \* dp[i - 1] + dp[i - 2]) % *mod*;  
 }  
 return (int)dp[n];  
}

5.Circle Printer

private static int count(String s) {  
 if (s == null || s.length() == 0) return 0;  
 String prev = "A", curr = prev;  
 int cnt = 0, d = 0;  
 for (char c : s.toCharArray()) {  
 curr = String.*valueOf*(c);  
 d = Math.*abs*(curr.compareTo(prev));  
 cnt += Math.*min*(d, 26 - d);  
 prev = curr;  
 }  
 return cnt;  
}

6.shared interest

public static int shared(int friendsNodes, List<Integer> from, List<Integer> to, List<Integer> w) {  
 Map<Integer,List<int[]>> map = new HashMap<>();  
 for (int i = 0; i < w.size(); ++i) {  
 int ff = from.get(i);  
 int tf = to.get(i);  
 map.computeIfAbsent(w.get(i), k -> new ArrayList<>()).add(new int[]{ff, tf});  
 }  
 int[][] cnt = new int[101][101];  
 int num = 0, res = 0;  
 for (int key: map.keySet()) {  
 List<int[]> list = map.get(key);  
 int[] friends = new int[101];  
 for (int i = 0; i < friends.length; ++i) {  
 friends[i] = i;  
 }  
 for (int i = 0; i < list.size(); ++i){  
 int[] e = list.get(i);  
 int root0 = *find*(friends, e[0]);  
 int root1 = *find*(friends, e[1]);  
 if (root1 != root0) {  
 friends[root1] = root0;}  
 }  
  
 for (int i = 0; i < friends.length; ++i) {  
 for (int j = i + 1; j < friends.length; ++j) {  
 if (*find*(friends,friends[i]) == *find*(friends, friends[j])) {  
 ++cnt[i][j];  
 if (cnt[i][j] > num) {  
 num = cnt[i][j];  
 res = i \* j;  
 } else if (cnt[i][j] == num) {  
 res = Math.*max*(res, i \* j);  
 }  
 }  
 }  
 }  
 }  
 return res;  
}

private static int find(int[] parent, int x) {  
 if (parent[x] != x) {  
 parent[x] = *find*(parent, parent[x]);  
 }  
 return parent[x];  
}

lc 780. 2055. 1381. 1507