**Laboratory work #2. Time Complexity. Sorting.**

Solve this problems using sorting algorithms and define their time, space complexity.

Deadline: 15th September 2020 Week 3

<https://leetcode.com/problems/increasing-decreasing-string/>



class Solution {

public String sortString(String s) {

int[] freq = new int[128];

for (char c : s.toCharArray()) {

freq[c]++;

}

int temp = s.length();

StringBuilder answer = new StringBuilder();

while (temp > 0) {

for (int i = 97; i < 123; i++) {

if (freq[i] > 0) {

freq[i]--;

temp--;

answer.append((char) i);

}

}

for (int i = 123; i > 96; i--) {

if (freq[i] > 0) {

freq[i]--;

temp--;

answer.append((char) i);

}

}

}

return answer.toString();

}

}

Our hashmap for getting frequency. ASCII for a to z is between 97 to 112. The cycle is populate our hashmap. The loop until all char of string are added are added //step 7. Then go from smallest a (97) to 123(z) // step 1,2 and 3. Then the loop go reverse from z to a // step 4, 5 and 6.

<https://leetcode.com/problems/average-salary-excluding-the-minimum-and-maximum-salary/>

class Solution {

public double average(int[] salary) {

Arrays.sort(salary);

double sum=0;

for(int i = 1; i < salary.length - 1; ++i){

sum += salary[i];

}

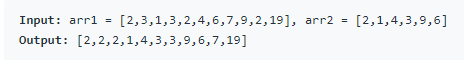
return sum / (salary.length-2);

}

}

The program firstly sorted the array, then loop start from 1 to the length -1. Because the minimum and maximum is not including to the average. Time complexity is O(n).

<https://leetcode.com/problems/relative-sort-array/>



class Solution {

public int[] relativeSortArray(int[] arr1, int[] arr2) {

int[] result = new int[arr1.length];

int counter = 0;

Arrays.sort(arr1);

for(int i = 0; i < arr2.length; i++){

for(int j = 0; j < arr1.length; j++){

if(arr2[i] == arr1[j]){

result[counter] = arr2[i];

counter++;

}

}

}

int m = result.length;

for(int i = 0; i < arr1.length; i++){

boolean flag = false;

for(int j = 0; j < arr2.length; j++){

if(arr1[i] == arr2[j]){

flag = true;

break;

}

}

if(!flag){

result[counter] = arr1[i];

counter++;

}

}

return result;

}

}

Time complexity is O(nm)

The program sort the the first array. Then the first loop run to the arr2 and the next loop run the arr1. If arr2 elements equal to the arr1 element, the arr2 element add to the result array. The next loops use to add tht missing elemenyt from arr1.

<https://leetcode.com/problems/sort-the-matrix-diagonally/>

class Solution {

public int[][] diagonalSort(int[][] mat) {

for(int i = 0; i < mat.length - 1; i++){

for(int j = 0; j < mat.length - 1; j++){

for(int n = 0; n < mat[j].length - 1; n++){

if(mat[j][n] > mat[j+1][n+1]){

int temp = mat[j][n];

mat[j][n] = mat[j+1][n+1];

mat[j+1][n+1] = temp;

}

}

}

}

return mat;

}

}

Time complexity is O(n^3)

The program runs for 3 loops. The second and third loop it to swap the elements in the diagonal. The first loop it to complete the all arrays.

<https://leetcode.com/problems/maximum-number-of-coins-you-can-get/>



class Solution {

public int maxCoins(int[] piles) {

int n = piles.length;

Arrays.sort(piles);

int result = 0;

for (int i=n/3; i<n; i+=2) {

result += piles[i];

}

return result;

}

}

Time complexity is O(nlogn)

The program sort the piles. Then loop start from length of pile divide by 3 to length of piles. But the i increased by 2 times. The result array add the piles I’s element.

<https://leetcode.com/problems/sort-integers-by-the-power-value/>

class Solution {

private static int power(int n){

if(n==1)return 0;

if(n%2==1)return 1+power(3\*n+1);

return 1+power(n/2);

}

public static int getKth(int lo, int hi, int k) {

List <Integer>ans[]=new ArrayList[1111];

for(int i=lo;i<=hi;i++){

int pow=power(i);

if(ans[pow]==null)

ans[pow]=new ArrayList<>();

ans[pow].add(i);

}

for(List<Integer> i:ans){

if(i!=null){

for(int e:i){

k--;

if(k<=0)return e;

}

}

}

return 0;

}

}

Time complexity is O(n^2)

The function calculates the power of the number. Create the list ans. The loop runs from lo to hi and ans[pow] add to the new arraylist. The next loop populates the list elements to the I and the i not equal to the null.

<https://leetcode.com/problems/largest-perimeter-triangle/>

class Solution {

public int largestPerimeter(int[] A) {

Arrays.sort(A);

for (int i = A.length - 3; i >= 0; --i)

if (A[i] + A[i+1] > A[i+2])

return A[i] + A[i+1] + A[i+2];

return 0;

}

}

Time complexity is O(nlogn)

The array A sorted. Then loop runs from the length of the A array -3 to the 0. If array I plus i+1 more than i+2 the program return the sum of the 3 elements of the array.

<https://leetcode.com/problems/intersection-of-two-arrays/>

class Solution {

public int[] intersection(int[] nums1, int[] nums2) {

Set<Integer> set = new HashSet<>();

for (int i : nums1) {

set.add(i);

}

List<Integer> result = new ArrayList<>();

for (int i : nums2) {

if(set.contains(i)){

result.add(i);

set.remove(i);

}

}

int[] ints = new int[result.size()];

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

ints[i]=result.get(i);

}

return ints;

}

}

Time complexity is O(nlogn)

Create the new set hashset and add to the set elements of the array nums1. Then create the array list to the adding the result. The loop populate the nums2 arrays element if set contains the I, the result list added the I element. Then the run loop to convert to the int elements.

<https://leetcode.com/problems/k-closest-points-to-origin/>

class Solution {

public int[][] kClosest(int[][] points, int K) {

int N = points.length;

int[] dists = new int[N];

for (int i = 0; i < N; ++i)

dists[i] = dist(points[i]);

Arrays.sort(dists);

int distK = dists[K-1];

int[][] ans = new int[K][2];

int t = 0;

for (int i = 0; i < N; ++i)

if (dist(points[i]) <= distK)

ans[t++] = points[i];

return ans;

}

public int dist(int[] point) {

return point[0] \* point[0] + point[1] \* point[1];

}

Time complexity is O(nlogn)

The array dists add the distance of the elements of dist. Array sort the dists array. The loop run from 0 to length of points array. The condition is the distance from the dists array more than distance from k, the points I’s add to the ans array.

<https://leetcode.com/problems/largest-number/>



class Solution {

public String largestNumber(int[] nums) {

String str[] = new String[nums.length];

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

str[i] = Integer.toString(nums[i]);

}

Arrays.sort(str, (a,b) ->{

return (b+a).compareTo(a+b);

});

String result = "";

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

result += str[i];

}

if(result.charAt(0) == '0'){

return "0";

}

return result;

}

}

Time complexity is O(nlogn)

The program create the new string, then loop runs convert the each elements of the nums and add to str string. Array sort the str and compare the place of the digit. The result string save the str I’s element to the string. If the result array the chat at equal to the 0 the program return 0.