**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 str) {

StringBuilder sb = new StringBuilder("");

int[] map = new int[26];

for (char ch : str.toCharArray())

map[ch - 'a']++;

boolean flag = true;

while (flag) {

flag = false;

for (int i = 0; i < 26; i++) {

if(map[i]!=0) {

sb.append((char)(i+'a'));

map[i]--;

flag = true;

}

}

for (int i = 25; i >= 0; i--) {

if(map[i]!=0) {

sb.append((char)(i+'a'));

map[i]--;

flag = true;

}

}

}

return sb.toString();

}

}

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

class Solution {

public double average(int[] salary) {

double result = 0;

int min = salary[0];

int max = salary[0];

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

if(salary[i] < min)

min = salary[i];

}

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

if(salary[i] > max)

max = salary[i];

}

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

if(salary[i] == min || salary[i] == max)

result += 0;

else

result += salary[i];

}

return result / (salary.length - 2);

}

}

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

class Solution {

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

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

int temp = 0;

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

for(int n : arr2){

set.add(n);

}

//Add all elements from arr2

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

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

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

result[temp] = arr2[i];

temp++;

}

}

}

// add rest of elements from arr1;

Arrays.sort(arr1);

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

if(!set.contains(arr1[i])){

result[temp] = arr1[i];

temp++;

}

}

return result;

}

}

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

class Solution {

public:

vector<vector<int>> diagonalSort(vector<vector<int>>& mat) {

for(int i=0; i < mat[0].size(); i++){

vector<int> tmp;

int x = 0;

int y = i;

while( x < mat.size() && y < mat[0].size()){

tmp.push\_back(mat[x][y]);

x+=1;

y+=1;

}

sort(tmp.begin(),tmp.end());

x = 0;

y = i;

for(auto el: tmp){

mat[x][y] = el;

x+=1;

y+=1;

}

}

for(int i=0; i < mat[0].size(); i++){

vector<int> tmp;

int x = i;

int y = 0;

while( x < mat.size() && y < mat[0].size()){

tmp.push\_back(mat[x][y]);

x+=1;

y+=1;

}

sort(tmp.begin(),tmp.end());

x = i;

y = 0;

for(auto el: tmp){

mat[x][y] = el;

x+=1;

y+=1;

}

}

return mat;

}

};

<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;

}

}

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

class Solution {

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

Map<Integer, Integer> map=new HashMap();

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

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

arr.add(i);

int p=getPower(i);

map.put(i, p);

}

Collections.sort(arr, (a, b) -> (map.get(a)==map.get(b))? a-b : map.get(a)-map.get(b));

return arr.get(k-1) ;

}

private int getPower(int n){

int count=0;

while(n>1){

count+=1;

if(n%2==0){

n=n/2;

}else{

n=3\*n+1;

}

}

return count;

}

}

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

class Solution {

public int largestPerimeter(int[] A) {

Arrays.sort(A);

for(int i=A.length-1; i>=0; i--){

if(i-2>=0){

int x=A[i],y=A[i-1],z=A[i-2];

if(x+y>z&&y+z>x&&z+x>y){

return x+y+z;

}

}

}

return 0;

}

}

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

class Solution {

public:

vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {

set<int> s(nums1.begin(), nums1.end()), res;

for (auto a : nums2) {

if (s.count(a)) res.insert(a);

}

return vector<int>(res.begin(), res.end());

}

};

Solution: you can sort the two arrays first, so you only need to traverse from the beginning of the two arrays to get the two elements num1 and num2.

if num1 == num2, then get the intersection elements, the two arrays get the next element respectively (as opposed to the previous element, excluding duplicate elements)

if num1> num2 then the second array gets the next element

if num1 <num2 then the first array gets the next element

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

class Solution {

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

PriorityQueue<Point> pq = new PriorityQueue<Point>((p1,p2) -> {

long distance = p2.getDistance()-p1.getDistance();

if(distance < 0) {

return -1;

} else if( distance > 0) {

return 1;

} else {

return 0;

}

});

for(int[] coord:points) {

int x= coord[0];

int y =coord[1];

long dist =x \* x + y \* y;

pq.add(new Point(dist,coord));

if(pq.size() > K) {

pq.poll();

}

}

int[][] out =new int[pq.size()][2];

int counter=0;

while(pq.size() > 0) {

out[counter++] = pq.poll().getCoords();

}

return out;

}

class Point {

long distance;

int[] coords;

public Point(long distance ,int[] coords) {

this.distance=distance;

this.coords=coords;

}

public long getDistance() {

return distance;

}

public int[] getCoords() {

return coords;

}

}

}

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

class Solution {

public:

string largestNumber(vector<int>& nums) {

string res;

sort(nums.begin(), nums.end(), [](int a, int b) {

return to\_string(a) + to\_string(b) > to\_string(b) + to\_string(a);

});

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

res += to\_string(nums[i]);

}

return res[0] == '0' ? "0" : res;

}

};