Amazon | OA 2020 | Nearest City

LEETCODE : <https://leetcode.com/discuss/interview-question/808374/>

Position: New Grad 2021

Amazon has Fulfillment Centers in multiple cities within a large geographic region. The cities are arranged on a group that has been divided up like an ordinary Cartesian plane. Each city is located at an integral(x,y) coordinate intersection. City names and locations are given in the form of three arrays: c,x, and y, which are aligned by the index to provide the city name (c[i]), and its coordinates, (x[i],y[i]).

**Write an algorithm to determine the name of the nearest city that shares an x or a y coordinate with the queried city. If no cities share an x or y coordinate, return none. If two cities have the same distance to the queried city, q[i], consider the one with an alphabetically smaller name (e.e 'ab' < 'aba' < 'abb') as the closest choice.**

The distance is denoted on a Euclidean plan: the difference in x plus the difference in y.

**Input**  
the input to the function/method consists of six arguments:  
***numOfCities***, an integer representing the number of cities;  
***cities***, a list of strings representing the names of each city[i];  
***xCoordinates***, a list of integers representing the X-coordinates of each city[i];  
***yCoordinates***, a list of integers representing the Y-coordinates of each city[i];  
***numOfQueries***, an integer representing the number of queries;  
***queries***, a list of strings representing the names of the queried cities.

**Output**  
Return a list of strings representing the name of the nearest city that shares either an x or a y coordinate with the queried city.

**Constraints**  
1 ≤ numOfCities, numOfQueries ≤ 10^5  
1 ≤ xCoordinates[i], yCoordinates[i] <= 10^9  
1 ≤ length of queries[i] and cities[i] ≤ 10

**Note**  
Each character of all c[i] and q[i] is in the range ascii[a-z, 0-9,-]  
All city name values, c[i] are unique. All cities have unique coordinates.

**Example:**

**Input:**

numOfCities = 3  
cities = ["c1", "c2", "c3"]  
xCoordinates = [3,2,1]  
yCoordinates = [3,2,3]  
numOfQueries = 3  
queries = ["c1", "c2", "c3"]

**Output:**

[c3, None, c1]

Ans: public class Main {

public static String[] findNearestCities(int numOfCities,String[] cities, int[] xCoordinates,int[] yCoordinates,int numOfQueries,

String[] queries) {

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

PriorityQueue<Map.Entry<String,Integer>> pq[] = new PriorityQueue[numOfCities];

HashMap<String,Integer> map1 = new HashMap<>();

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

map1.put(cities[i],i);

}

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

pq[i] = new PriorityQueue<>((a,b) -> a.getValue()-b.getValue()!=0?a.getValue()-b.getValue():a.getKey().compareTo(b.getKey()));

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

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

if(i == j)

continue;

if(xCoordinates[i] == xCoordinates[j]){

int x = Math.abs(xCoordinates[i] - xCoordinates[j]);

int y = Math.abs(yCoordinates[i] - yCoordinates[j]);

int d = x+y;

HashMap<String,Integer> m = new HashMap<>();

m.put(cities[j],d);

for(Map.Entry<String,Integer> e : m.entrySet())

pq[i].offer(e);

}

}

}

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

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

if(i == j)

continue;

if(yCoordinates[i] == yCoordinates[j]){

int x = Math.abs(xCoordinates[i] - xCoordinates[j]);

int y = Math.abs(yCoordinates[i] - yCoordinates[j]);

int d = x+y;

HashMap<String,Integer> m = new HashMap<>();

m.put(cities[j],d);

for(Map.Entry<String,Integer> e : m.entrySet())

pq[i].offer(e);

}

}

}

String res[] = new String[numOfQueries];

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

int idx = map1.get(queries[i]);

if(!pq[idx].isEmpty()){

Map.Entry<String,Integer> e = pq[idx].peek();

res[i] = e.getKey();

}

else res[i]= null;

}

return res;

}

public static void main(String[] args) {

int numOfCities = 6;

String[] cities = {"green","yellow","red","blue","grey","pink"};

int[] xCoordinates = {10,20,15,30,10,15} ;

int[] yCoordinates = {30,25,30,40,25,25};

int numOfQueries = 4;

String[] queries = {"grey","blue","red","pink"};

String res[] = findNearestCities(numOfCities,cities,xCoordinates,yCoordinates,numOfQueries,queries);

for(String s : res)

System.out.print(s + " ");

}

}

Alternate : public static void main(String[] args) {

String[] cities = { "c1", "c2", "c3" };

int[] xs = { 3, 2, 1 };

int[] ys = { 3, 2, 3 };

String[] queries = { "c1", "c2", "c3" };

System.out.println(Arrays.toString(solve(cities, xs, ys, queries)));

}

private static String[] solve(String[] cities, int[] xs, int[] ys, String[] queries) {

String[] res = new String[queries.length];

Map<Integer, TreeMap<Integer, String>> xMap = new HashMap<>();

Map<Integer, TreeMap<Integer, String>> yMap = new HashMap<>();

Map<String, int[]> nodeMap = new HashMap<>();

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

nodeMap.put(cities[i], new int[] {xs[i], ys[i]});

xMap.putIfAbsent(xs[i], new TreeMap<>());

yMap.putIfAbsent(ys[i], new TreeMap<>());

xMap.get(xs[i]).put(ys[i], cities[i]);

yMap.get(ys[i]).put(xs[i], cities[i]);

}

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

int[] node = nodeMap.get(queries[i]);

TreeMap<Integer, String> ym = xMap.getOrDefault(node[0], new TreeMap<>());

TreeMap<Integer, String> xm = yMap.getOrDefault(node[1], new TreeMap<>());

Integer yl = ym.lowerKey(node[1]), yh = ym.higherKey(node[1]);

Integer xl = xm.lowerKey(node[0]), xh = xm.higherKey(node[0]);

int dist = Integer.MAX\_VALUE;

if(yl != null && Math.abs(yl - node[1]) <= dist) {

dist = Math.abs(yl - node[1]);

res[i] = res[i] == null ? ym.get(yl) : res[i].compareTo(ym.get(yl)) > 0 ? ym.get(yl) : res[i];

}

if(yh != null && Math.abs(yh - node[1]) <= dist) {

dist = Math.abs(yh - node[1]);

res[i] = res[i] == null ? ym.get(yh) : res[i].compareTo(ym.get(yh)) > 0 ? ym.get(yh) : res[i];

}

if(xl != null && Math.abs(xl - node[0]) <= dist) {

dist = Math.abs(xl - node[0]);

res[i] = res[i] == null ? xm.get(xl) : res[i].compareTo(xm.get(xl)) > 0 ? xm.get(xl) : res[i];

}

if(xh != null && Math.abs(xh - node[1]) <= dist) {

dist = Math.abs(xh - node[1]);

res[i] = res[i] == null ? xm.get(xh) : res[i].compareTo(xm.get(xh)) > 0 ? xm.get(xh) : res[i];

}

if(res[i] == null)

res[i] = "None";

}

return res;

}