**Problem No. 1**

Three points on a plane are called collinear if they lie on the same straight line. For example, points with Cartesian coordinates (0,1), (1,3) and (3,7) are collinear. Points (0,0), (1,1) and (2,3) are not.

Assume that the following declarations are given:

*class Point2D {*

*public int x;*

*public int y;*

*}*

Write a function:

*class Solution {public solution (Point2D [] A);*

*}*

that, given an array A containing Cartesian coordinates of N distinct points, counts the number of triplets (A[P], A[Q], A[R]) such that 0<=P < Q < R < N and the points A[P], A[Q] and A[R] are collinear. The function should return -1 if the number of such triplets exceeds 100,000,000.

To access the coordinates of the K-th point (where 0<=K < N), use the following syntax:

* A[K]. x to access the x-coordinate.
* A[K]. y to access the y-coordinate.

For example, given A such that:

A [0].x = 0 A [0]. y = 0

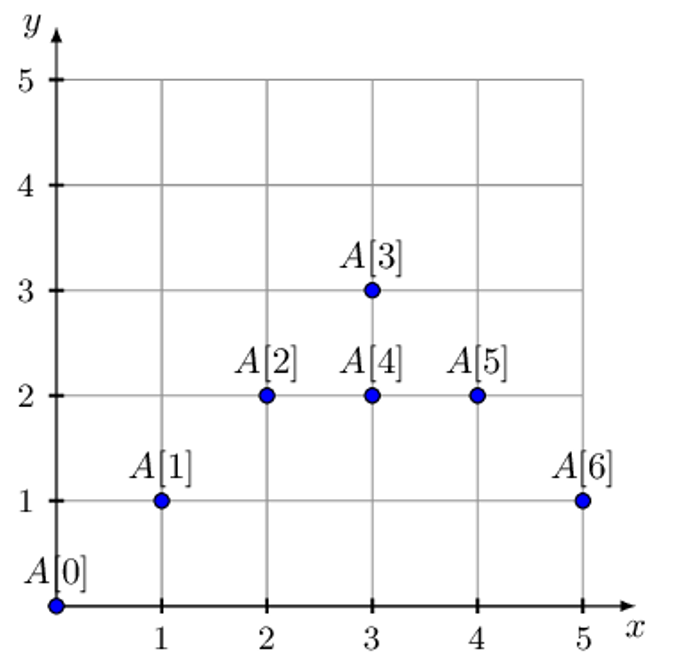
A [1].x = 1 A [0]. y = 1

A [2].x = 2 A [0]. y = 2

A [3].x = 3 A [0]. y = 3

A [4].x = 4 A [0]. y = 2

A [5].x = 5 A [0]. y = 2



the function should return 6, because the following six triplets of points are collinear:

(A [0], A [1], A [2])

(A [0], A [1], A [3])

(A [0], A [2], A [3])

(A [1], A [2], A [3])

(A [2], A [4], A [5])

(A [3], A [5], A [6])

Assume that:

* N is an integer within the range [1…1,000];
* the coordinates of each point in array A are integers within the range [0…9,999];
* the elements of A are all distinct.

Complexity:

* expected worst-case time complexity is O(N2\*log(N));
* expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

**Problem No. 2**

There is a queue of N cars waiting at a filling station. There are three fuel dispensers at the station, labeled X, Y and Z, respectively. Each dispenser has some finite amount of fuel in it; always, the amount of available fuel is clearly displayed on each dispenser.

When a car arrives at the front of the queue, the driver can choose to drive to any dispenser not occupied by another car. Suppose that the fuel demand is in D liters for this car, the driver must choose a dispenser which has at least D liters of fuel. If all unoccupied dispensers have less than D liters, the driver must wait for some other car to finish tanking up. If all dispensers are unoccupied, and none has at least D liters, the driver is unable to refuel the car and it blocks the queue indefinitely. If more than one unoccupied dispenser has at least D liters, the driver chooses the one labeled with the smallest letter among them.

Each driver will have to wait some amount of time before he or she starts refueling the car. Calculate the maximum waiting time among all drivers. Assume that tanking one liter of fuel takes exactly one second, and moving cars is instantaneous.

Write a function:

*class Solution {public int solution (int[] A, int X, int Y, int Z);*

*}*

that, given a zero-indexed array A consisting of N integers (which specify the fuel demands in liters for subsequent cars in the queue), and numbers X, Y and Z (which specify the initial amount of fuel in the respective dispensers) returns the maximum waiting time for the car. If any car is unable to refuel, the function should return -1.

For example, given X = 7, Y = 11, Z = 3 and the following array A:

A [0] = 2

A [1] = 8

A [2] = 4

A [3] = 3

A [4] = 2

the function should return 8. The subsequent cars will have to wait in the queue for 0, 0, 2, 2 and 8 seconds, respectively. The scenario is as follows:

* At time 0, car 0 drives to dispenser X.
* At time 0, car 1 drives to dispenser Y.
* There is not enough fuel in the dispenser Z to satisfy the demands of car 2, so this car must wait. At time 2 car 0 finishes refueling and car drives to dispenser X.
* At time 2 car 3 drives to dispenser Z.
* Now all dispensers are occupied, so car 4 waits. There will not be enough fuel in dispensers’ X and Z after cars 2 and 3 finish tanking up, so car 4 waits until car 1 finishes refueling at dispenser Y. At time 8, car 4 drives to dispenser Y.

For X = 4, Y = 0 and Z =3 and array A:

A [0] = 5

the function should return -1.

Assume that:

* N is an integer within the range [1…100,000];
* Each element of array A is an integer within the range [1…1,000,000,000];
* X, Y and Z are integers within the range [1…1,000,000,000];

Complexity:

* expected worst-case time complexity is O(N);
* expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input array can be modified.