An array A consisting of N integers is given. A triplet (P, Q, R) is *triangular* if 0 ≤ P < Q < R < N and:

* A[P] + A[Q] > A[R],
* A[Q] + A[R] > A[P],
* A[R] + A[P] > A[Q].

For example, consider array A such that:

A[0] = 10 A[1] = 2 A[2] = 5 A[3] = 1 A[4] = 8 A[5] = 20

Triplet (0, 2, 4) is triangular.

Write a function:

int solution(vector<int> &A);

that, given an array A consisting of N integers, returns 1 if there exists a triangular triplet for this array and returns 0 otherwise.

For example, given array A such that:

A[0] = 10 A[1] = 2 A[2] = 5 A[3] = 1 A[4] = 8 A[5] = 20

the function should return 1, as explained above. Given array A such that:

A[0] = 10 A[1] = 50 A[2] = 5 A[3] = 1

the function should return 0.

Write an **efficient** algorithm for the following assumptions:

* N is an integer within the range [0..100,000];
* each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].