Minimize Max Distance to Gas Station

We have a horizontal number line. On that number line, we have gas stations at positions stations[0], stations[1], ..., stations[N-1], where n = size of the stations array. Now, we add k more gas stations so that d, the maximum distance between adjacent gas stations, is minimized. We have to find the smallest possible value of d. Find the answer exactly to 2 decimal places.

Example 1:

Input:

n = 10

stations = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

k = 9

Output: 0.50

Explanation: Each of the 9 stations can be added mid way between all the existing adjacent stations.

Example 2:

Input:

n = 10

stations = [3,6,12,19,33,44,67,72,89,95]

k = 2

Output: 14.00

Explanation: Construction of gas stations at 8th(between 72 and 89) and 6th(between 44 and 67) locations.

Your Task:

You don't need to read input or print anything. Your task is to complete the function findSmallestMaxDist() which takes a list of stations and integer k as inputs and returns the smallest possible value of d. Find the answer exactly to 2 decimal places.

Expected Time Complexity: O(n\*log k)

Expected Auxiliary Space: O(1)

Constraint:

10 <= n <= 5000

0 <= stations[i] <= 109

0 <= k <= 105

stations is sorted in a strictly increasing order.

Code :

class Solution {

public:

int noOfGasStationRequired(vector<int> arr, int n, double dist) {

int cntStations = 0;

for(int i = 1;i < n; i++) {

int noInBetween = (arr[i]-arr[i-1])/dist;

cntStations += noInBetween;

}

return cntStations;

}

double findSmallestMaxDist(vector<int> &stations, int k) {

int n = stations.size();

double low = 0, high = 0;

for(int i = 0;i < n-1; i++)

high = max(high, (double)(stations[i+1]-stations[i]));

while(high-low > 1e-6) {

double mid = (low+high)/2;

if(noOfGasStationRequired(stations, n, mid) > k)

low = mid;

else

high = mid;

}

return high;

}

};