**1870. Minimum Speed to Arrive on Time :-**

Medium Accepted: 53.3K Submissions: 120.5K Acceptance Rate:44.2%

You are given a floating-point number hour, representing the amount of time you have to reach the office. To commute to the office, you must take n trains in sequential order. You are also given an integer array dist of length n, where dist[i] describes the distance (in kilometers) of the ith train ride.

Each train can only depart at an integer hour, so you may need to wait in between each train ride.

* For example, if the 1st train ride takes 1.5 hours, you must wait for an additional 0.5 hours before you can depart on the 2nd train ride at the 2 hour mark.

Return *the****minimum positive integer****speed****(in kilometers per hour)****that all the trains must travel at for you to reach the office on time, or*-1*if it is impossible to be on time*.

Tests are generated such that the answer will not exceed 107 and hour will have **at most two digits after the decimal point**.

**Example 1:**

**Input:** dist = [1,3,2], hour = 6

**Output:** 1

**Explanation:** At speed 1:

- The first train ride takes 1/1 = 1 hour.

- Since we are already at an integer hour, we depart immediately at the 1 hour mark. The second train takes 3/1 = 3 hours.

- Since we are already at an integer hour, we depart immediately at the 4 hour mark. The third train takes 2/1 = 2 hours.

- You will arrive at exactly the 6 hour mark.

**Example 2:**

**Input:** dist = [1,3,2], hour = 2.7

**Output:** 3

**Explanation:** At speed 3:

- The first train ride takes 1/3 = 0.33333 hours.

- Since we are not at an integer hour, we wait until the 1 hour mark to depart. The second train ride takes 3/3 = 1 hour.

- Since we are already at an integer hour, we depart immediately at the 2 hour mark. The third train takes 2/3 = 0.66667 hours.

- You will arrive at the 2.66667 hour mark.

**Example 3:**

**Input:** dist = [1,3,2], hour = 1.9

**Output:** -1

**Explanation:** It is impossible because the earliest the third train can depart is at the 2 hour mark.

**Constraints:**

* n == dist.length
* 1 <= n <= 105
* 1 <= dist[i] <= 105
* 1 <= hour <= 109
* There will be at most two digits after the decimal point in hour.

**Code :-**

class Solution {

public:

    bool check(vector<int> &dist, double hour, int sp){

        int n=dist.size();

        double temphour=0;

        for(auto i=0; i<n; i++){

            temphour += (double)dist[i] / sp;

            //cout<<"("<<((double)dist[i] / sp)<<",";

            if(i != n-1)

                temphour = ceil(temphour);

            //cout<<temphour<<")";

        }

        return temphour <= hour;

    }

    int minSpeedOnTime(vector<int>& dist, double hour) {

        int n=dist.size(), ans=-1;

        if(hour <= n-1)     return ans;

        int high=INT\_MAX, low=1, mid;

        while(low <= high){

            mid = low + (high-low)/2;

            if(check(dist, hour, mid)){

                ans = mid;

                high = mid - 1;

            }

            else

                low = mid + 1;

        }

        return ans;

    }

};

**T.C :- O(N\*log N)**

**S.C :- O(1) without recursion call stack**