KOKO EATING BANANAS

✩ In this problem, we are given an array of integers where each integer represents the number of bananas in a pile. Our job is to give how many bananas KOKO should eat per hour to finish all piles under "N" hours.

Brute force solution is to check for each number from 1 to ∞ until the total number of hours falls under our deadline.

Maximum value for our answer is same as maximum element in the array, as then each pile is eaten in 1 hour.

So we can apply binary search from 1 to Maximum, where we eliminate the right half if our number of hours taken is less than the deadline and left half if our number of hours taken is more than the deadline.

Pseudocode :

```
kokoEatingBananas(arr, N, K) {
    max = INT_MIN
    for (i = 0 → N)  {
        if (arr [i] > max)  {
            max = arr [i]
        }
    }
    low = 1
    high = max
    ans = max
    while (low <= high)  {
        mid = (low + high) / 2
        h = hours (arr , N , mid)
        if (h == K)  {
            return mid
        } else if (h < K)  {
            ans = mid
            low = mid + 1
        } else  {
            high = mid - 1
        }
    }
    return ans
}

hours (arr , N , a)  {
    int h = 0
    for (i = 0 → N)  {
        h + = arr [i] / a
    }
    return h
}
```