# **Leetcode 875 – Koko Eating Bananas**

## Problem Understanding

Koko loves bananas and wants to eat all piles in h hours.

* She can eat from **one pile per hour** at a fixed speed k (bananas/hour).
* If a pile has less than k, she eats the whole pile.
* She **can't switch between piles in the same hour**.

**Goal:** Find the **minimum integer k** such that Koko can eat all bananas within h hours.

## Optimized Java Solution (Binary Search on Answer)

class Solution {

public int minEatingSpeed(int[] piles, int h) {

int left = 1;

int right = getMax(piles);

int result = right;

while (left <= right) {

int mid = left + (right - left) / 2; // prevent overflow

if (canFinish(piles, h, mid)) {

result = mid;

right = mid - 1; // Try smaller k

} else {

left = mid + 1; // Need larger k

}

}

return result;

}

private boolean canFinish(int[] piles, int h, int k) {

int hours = 0;

for (int pile : piles) {

hours += (pile + k - 1) / k; // Same as ceil(pile / k)

if (hours > h) return false; // Early exit

}

return true;

}

private int getMax(int[] piles) {

int max = 0;

for (int pile : piles) {

max = Math.max(max, pile);

}

return max;

}

}

## Dry Run Using Table

Input:

piles = [3, 6, 7, 11], h = 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iteration | Left | Right | Mid | Total Hours | Result | Action |
| 1 | 1 | 11 | 6 | 8 | 6 | Try smaller k |
| 2 | 1 | 5 | 3 | 12 | 6 | Increase k |
| 3 | 4 | 5 | 4 | 10 | 6 | Increase k |
| 4 | 5 | 5 | 5 | 9 | 5 | Try smaller k |
| 5 | 5 | 4 | — | — | 6 | End loop |

**Final answer:** 6

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(n × logM) |
| Space | O(1) |

* n = number of piles
* M = max value in piles (range of k values)

## Alternate Approaches

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Time | Space | Notes |
| ✅ Binary Search | O(n logM) | O(1) | Best choice |
| ❌ Brute Force | O(n × M) | O(1) | TLE for large piles |