Koko Eating Bananas - Approaches and Code

Problem Summary

Problem Summary:

Koko loves eating bananas. There are piles[] of bananas and an integer h (number of hours).

Koko can eat k bananas per hour. If a pile has less than k bananas, she eats all of it in 1 hour.

Objective: Find the minimum integer k such that Koko can eat all the bananas in h hours.

Approach 1: Brute Force

Approach 1: Brute Force (TLE for large inputs)

- Try all k from 1 to max(piles).

- For each k, calculate total hours required.

- Return the smallest k such that total hours <= h.

Time Complexity: O(max(piles) * n)

Approach 2: Binary Search

Approach 2: Binary Search (Efficient)

Why Binary Search?

- The answer lies in a range -> k in [1, max(piles)].

- The function isValid(k): "Can Koko eat all in h hours if she eats k bananas/hour?" is monotonic.

Steps:

1. Set low = 1, high = max(piles).

2. Perform binary search:

Koko Eating Bananas - Approaches and Code

- Mid = (low + high) / 2.

- Calculate total hours needed at speed = mid.

- If hours <= h: store mid in ans and try smaller speeds (high = mid - 1).

- Else: increase speed (low = mid + 1).

Time Complexity: O(n * log(max(piles)))

Space Complexity: O(1)

Your Code

```
class Solution {
public:
    int minEatingSpeed(vector<int>& piles, int h) {
        int maxele = *max_element(piles.begin(), piles.end());
        int st = 1, end = maxele;
        int ans = INT_MAX;

        while (st <= end) {
            int mid = st + (end - st) / 2;
            int count = 0;
            for (int pile : piles) {
                count += (pile + mid - 1) / mid;  // Faster than ceil()
            }

            if (count <= h) {
                ans = min(ans, mid);
                end = mid - 1;
            } else {
                st = mid + 1;
            }
        }

        return ans;
    }
};
```