Questions on Binary search:

Problem Statement: You are given a positive integer n. Your task is to find and return its square root. If ‘n’ is not a perfect square, then return the floor value of 'sqrt(n)'.

Note: *The question explicitly states that if the given number, n, is not a perfect square, our objective is to find the maximum number, x, such that x squared is less than or equal to n (x*x <= n). In other words, we need to determine the floor value of the square root of n.*

Examples

```
Example 1:Input Format: n = 36
Result: 6
Explanation: 6 is the square root of 36.

Example 2:Input Format: n = 28
Result: 5

Explanation: Square root of 28 is approximately 5.292. So, the
floor value will be 5.
```

```
#include <bits/stdc++.h>
using namespace std;

int floorSqrt(int n) {
    int low = 1, high = n;
    //Binary search on the answers:
    while (low <= high) {
        long long mid = (low + high) / 2;
        long long val = mid * mid;
        if (val <= (long long)(n)) {
            //eliminate the left half:
            low = mid + 1;
        }
        else {
            //eliminate the right half:
            high = mid - 1;
        }
    }
    return high;
}

int main()
{
    int n = 28;
    int ans = floorSqrt(n);
    cout << "The floor of square root of " << n
         << " is: " << ans << "\n";
    return 0;
}
```

https://leetcode.com/problems/koko-eating-bananas/

```
#include <bits/stdc++.h>
using namespace std;

int findMax(vector<int> &v) {
    int maxi = INT_MIN;
    int n = v.size();
    //find the maximum:
    for (int i = 0; i < n; i++) {
        maxi = max(maxi, v[i]);
    }
    return maxi;
}

int calculateTotalHours(vector<int> &v, int hourly) {
    int total = 0;
    int n = v.size();
    //find total hours:
    for (int i = 0; i < n; i++) {
```

```cpp
        total += ceil((double)(v[i]) / (double)(hourly));
    }
    return total;
}

int minimumRateToEatBananas(vector<int> v, int h) {
    int l = 1, r = findMax(v);

    //apply binary search:
    while (l <= r) {
        int mid = (l+r) / 2;
        int total = calculateTotalHours(v, mid);
        if (total <= h) {
            r = mid - 1;
        }
        else {
            l = mid + 1;
        }
    }
    return l;
}

int main()
{
    vector<int> v = {7, 15, 6, 3};
    int h = 8;
    int ans = minimumRateToEatBananas(v, h);
    cout << "Koko should eat atleast " << ans << " 
bananas/hr.\n";
    return 0;
}
```