**🧠 First, What Are We Solving?**

You are doing **binary search**, and in each step, you cut the array size in half:

* Start with n elements
* After 1 step → n/2
* After 2 steps → n/4 (which is n / 2²)
* After 3 steps → n/8 (which is n / 2³)
* ...
* After k steps → n / 2^k

You stop when you are left with **only 1 element**.

So the idea is:

n2k=1\frac{n}{2^k} = 12kn​=1

We’re trying to find **how many steps (k)** it takes to reach 1.

**👣 Step-by-Step with Example: n = 8**

Let’s try manually:

| **Step** | **Elements Remaining** | **Math Used** |
| --- | --- | --- |
| 0 | 8 | 8 / 2⁰ = 8 |
| 1 | 4 | 8 / 2¹ = 4 |
| 2 | 2 | 8 / 2² = 2 |
| 3 | 1 | 8 / 2³ = 1 ✅ |

So you stopped at:

823=1⇒23=8⇒k=3\frac{8}{2^3} = 1 \Rightarrow 2^3 = 8 \Rightarrow \boxed{k = 3}238​=1⇒23=8⇒k=3​

**🔄 Now Flip the Equation**

We had:

n2k=1\frac{n}{2^k} = 12kn​=1

Multiply both sides by 2k2^k2k:

n=2kn = 2^kn=2k

Now we ask:  
**"2 to the power what gives n?"**  
That’s exactly what logarithm answers!

k = \log\_2(n)  
]