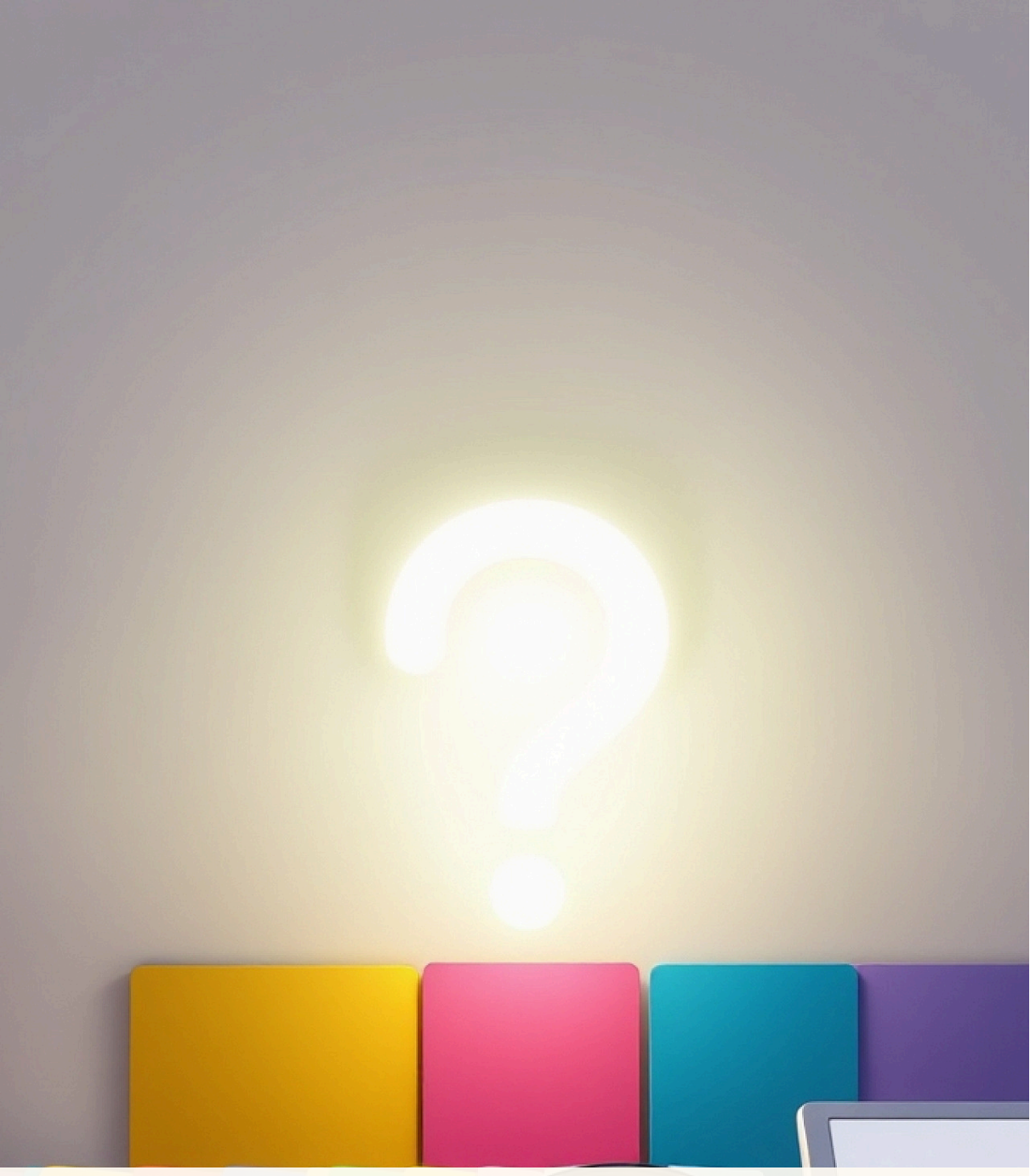
How does binary search work?



www.yzokul.com.tr

How does binary search work?



You want to know how binary search works. The short answer is that it checks the middle item of sorted data and keeps halving the area until it finds the target. This method is fast because it throws away half of the remaining items with each check. It only works correctly when the data is sorted in order. You will learn each part of this process in the following pages.



Picture your data as items lined up in a row, like numbers on cards placed side by side. This row is the group you will search through. Each card holds one value, and the cards are arranged in a fixed order. You can point to any card by its position in the row. The goal is to find a specific target value among these cards.

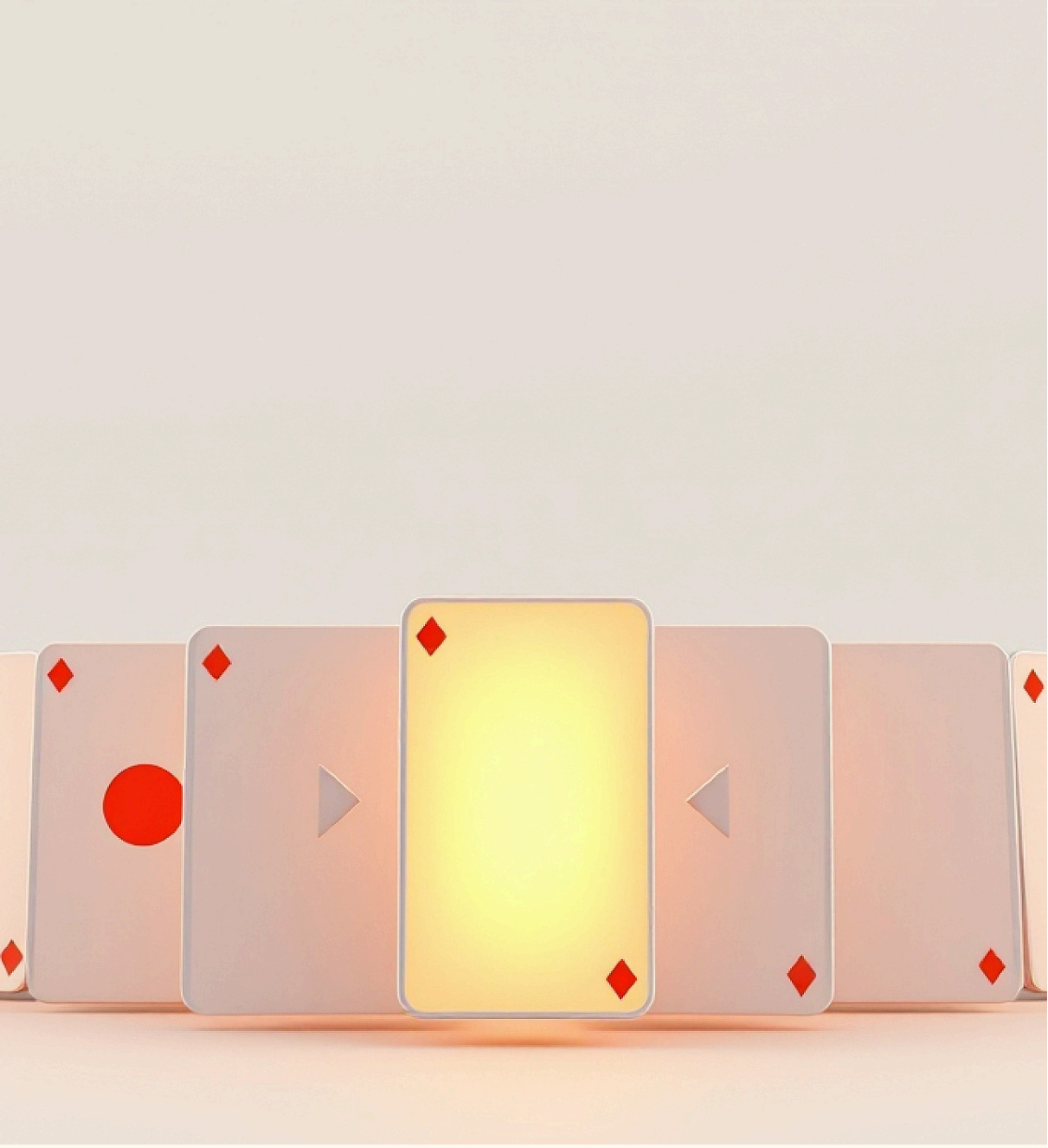




For binary search to work, the row must be in order, for example from smallest to largest. This order is called sorted data. Without this order, the halving trick cannot work correctly because you cannot know which half contains the target. Always check that your data is sorted before using binary search. If it is not sorted, sort it first.



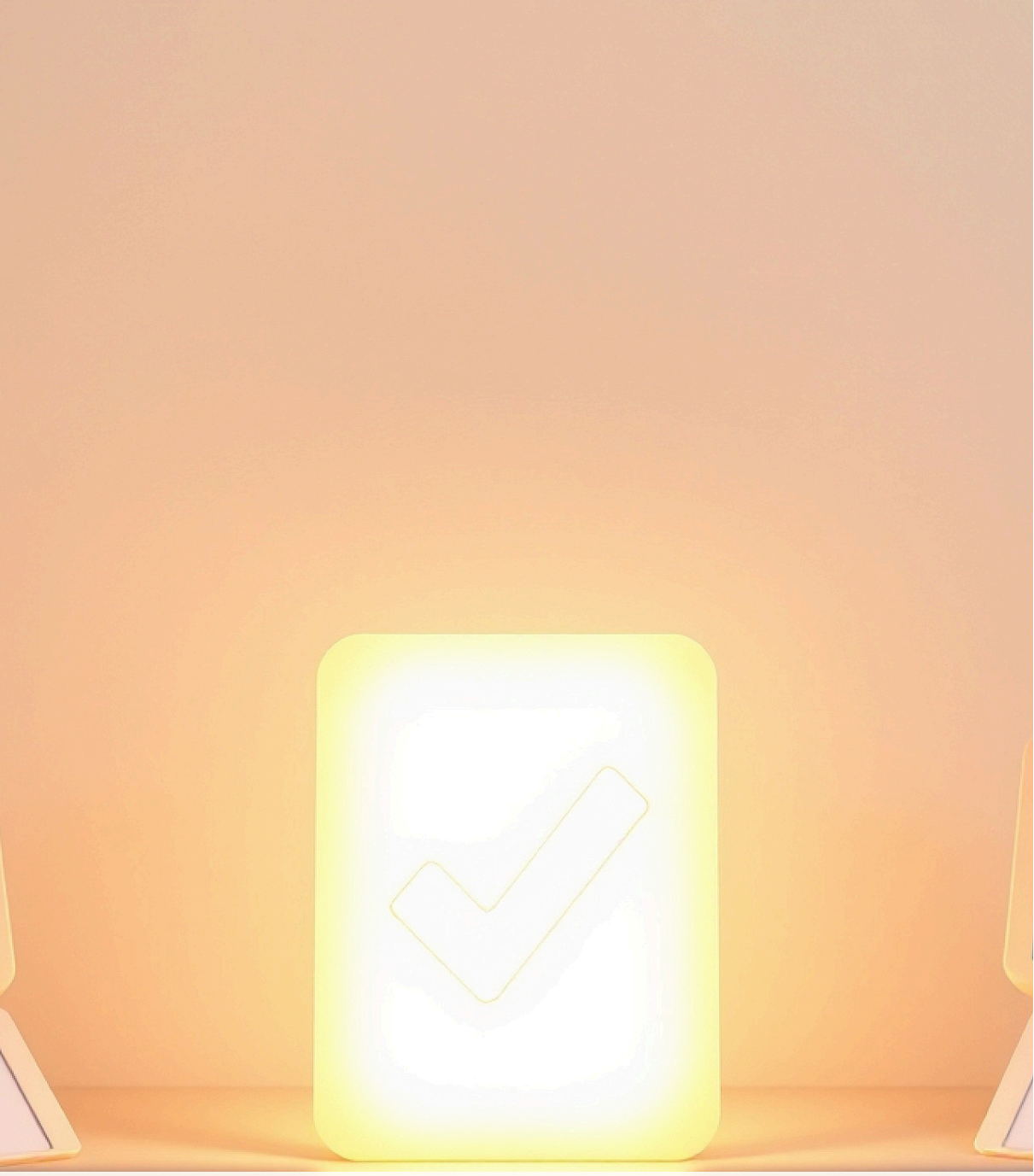
Look at a concrete sorted row: 2, 5, 8, 12, 16, 23, 38. These numbers are in increasing order. Suppose you want to find the number 23. This row is your search area. You will use binary search to locate 23 quickly. Notice that the numbers are evenly spaced in value, but that is not required—only the order matters.



To find the middle item, count the items and pick the one in the center. In the row 2, 5, 8, 12, 16, 23, 38, there are seven items, so the middle is the fourth item: 12. That middle card is the first one binary search looks at. If there is an even number of items, you can pick either of the two middle ones; the method still works. The middle item divides the row into a left part and a right part.



Compare your target (23) with the middle value (12). The answer is one of three things: equal, smaller, or larger. Here, 23 is larger than 12. This comparison tells you which half to keep. You do not need to check every item; the comparison guides you. The result of this comparison decides your next move.



If the target equals the middle value, the search is done. You have found the item in a single check. For example, if you were searching for 12 in the row, you would stop immediately because 12 is the middle. This is the best case: the target is exactly in the middle. Then you can report that the item is found.



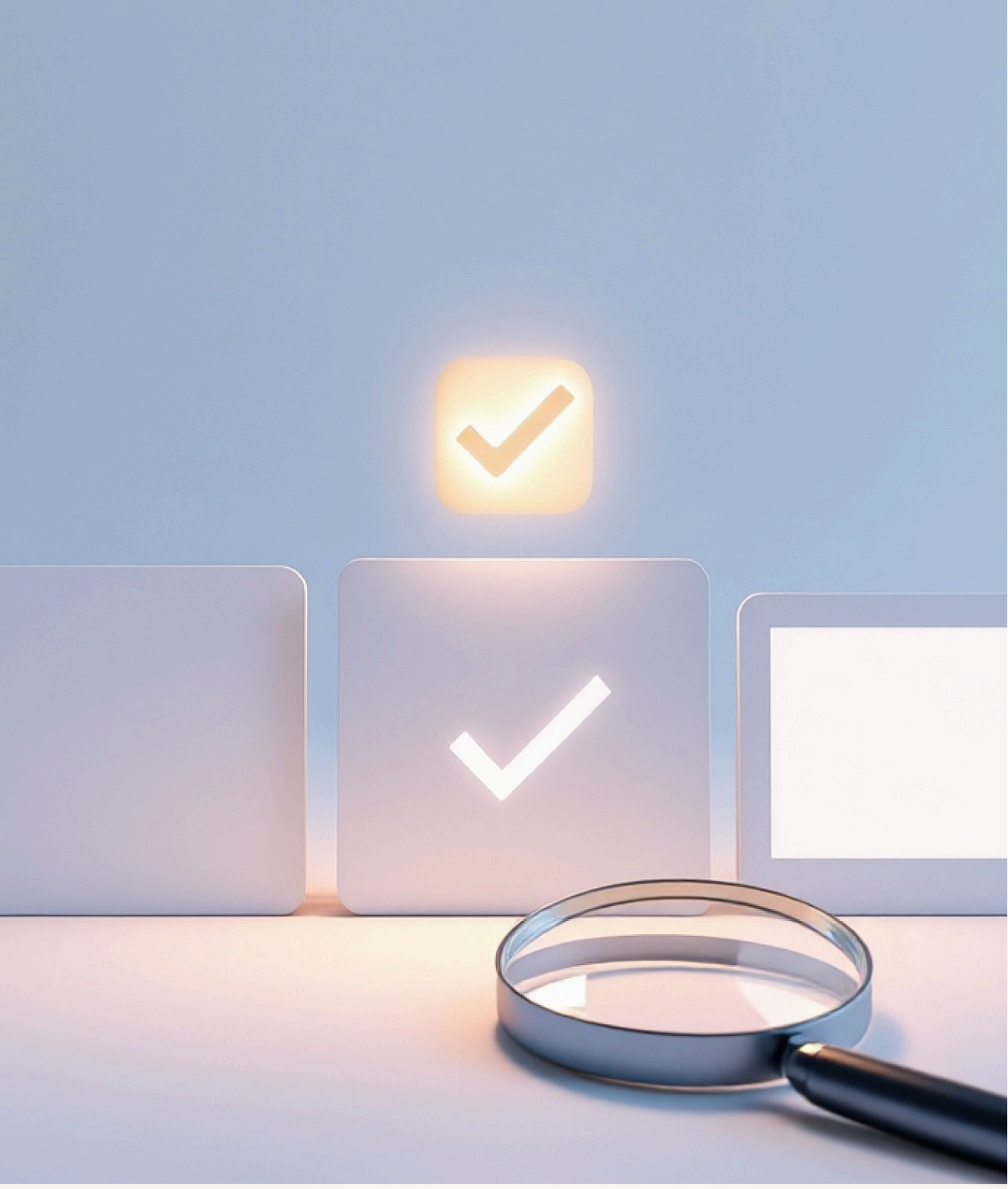
If the target is smaller than the middle value, you ignore the middle and everything to its right. You keep only the left half. For instance, if you were searching for 8 in the row, 8 is smaller than 12, so you discard 12 and all numbers to its right (16, 23, 38). The new search area is 2, 5, 8. You then repeat the process on this smaller row.



If the target is larger than the middle value, you ignore the middle and everything to its left. You keep only the right half. In our example, 23 is larger than 12, so you discard 12 and all numbers to its left (2, 5, 8). The new search area is 16, 23, 38. You then repeat the process on this smaller row.



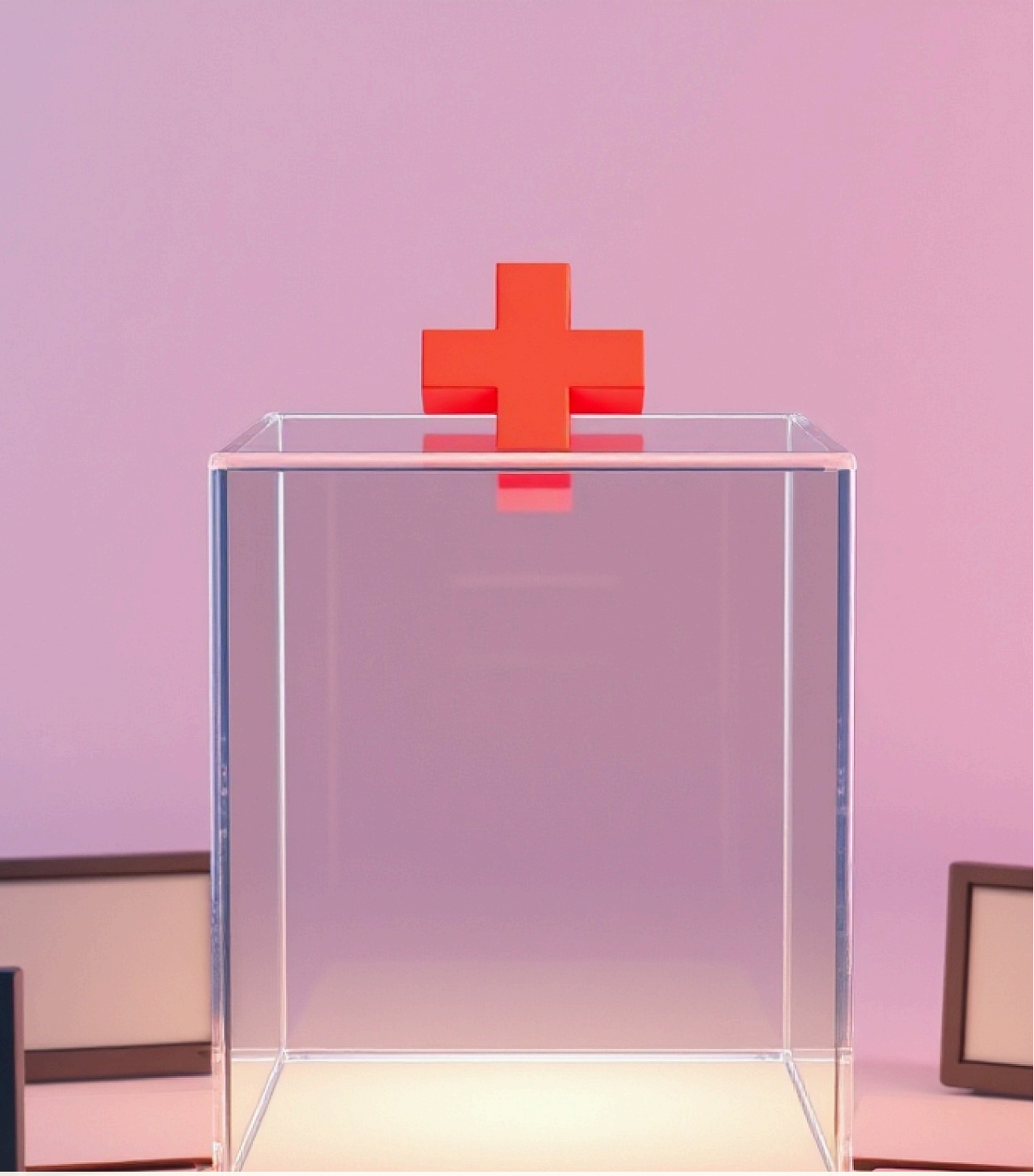
Each check throws away about half of the remaining items. The candidate area shrinks quickly with every step. For example, starting with 7 items, after one check you have at most 3 items left. After two checks, at most 1 item remains. This halving is why binary search is fast, even for large data sets. The number of steps needed is small compared to checking every item.



Take the remaining half and find its middle item again. In the new row 16, 23, 38, the middle is 23. Compare your target (23) with 23: they are equal, so you have found it. If they were not equal, you would again keep the appropriate half. You repeat this process until you find the target or run out of items.



Think of finding a word in a dictionary. You open the dictionary near the middle and see if your word comes before or after that page. Then you take the appropriate half and open near its middle, repeating until you find the word. This is exactly how binary search works: each comparison tells you which half to search next. The dictionary is sorted alphabetically, just like your data must be sorted.



The search stops when the middle value matches the target. It also stops when no items are left, which means the target is not there. For example, if you search for 7 in the row 2, 5, 8, 12, 16, 23, 38, you will eventually narrow down to an empty area and conclude that 7 is absent. Knowing when to stop is important to avoid infinite loops.



A common mistake is using binary search on data that is not sorted. The halves then do not contain what you expect, and the answer can be wrong. For example, if the row were 12, 2, 38, 5, 16, 8, 23, the middle is 5, but the left half is not all smaller than 5. Always ensure your data is sorted before applying binary search.



Another error is jumping to a half without comparing to the middle first. The comparison is what tells you which half to keep. If you skip it, you might choose the wrong half and miss the target. Always compare the target with the middle value before deciding which half to discard. This simple check is the heart of the method.



Binary search checks the middle of sorted data, then keeps the half where the target must be. The area halves each step, so it finds items fast. Remember to sort your data first and always compare with the middle. With practice, you can apply this method to any sorted list. It is a powerful tool for efficient searching.