The Design and Analysis of Algorithm

Binary search

Instructor: Sadullah Karimi

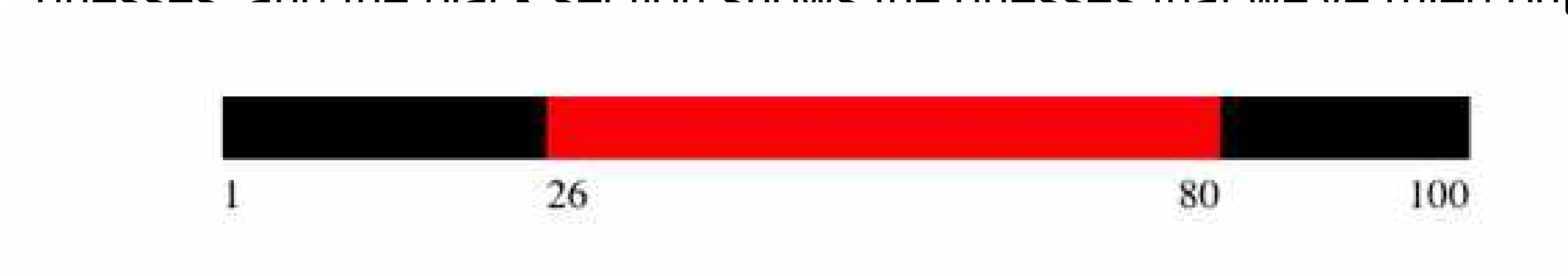
M.Sc. in CSE

Binary Search

- Binary search is an efficient algorithm for finding an item from a sorted list of items.
- It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one.
- We used binary search in the guessing game in the introductory tutorial.

Describing binary search

The main idea of binary search is to keep track of the current range of reasonable guesses. Let's say that I'm thinking of a number between one and 100, just like the guessing game. If you've already guessed 25 and I told you my number was higher, and you've already guessed 81 and I told you my number was lower, then the numbers in the range from 26 to 80 are the only reasonable guesses. Here, the red section of the number line contains the reasonable guesses, and the black section shows the guesses that we've ruled out:



Here's a step-by-step description of using binary search to play the guessing game:

- Let min = 1 and max = n
- Guess the average of max and min, rounded down so that it is an integer.
- If you guessed the number, stop. You found it!
- If the guess was too low, set min to be one larger than the guess.
- If the guess was too high, set max to be one smaller than the guess.
- Go back to step two.

Implementing binary search of an array

Here's a python array of the first 25 prime numbers,in order:

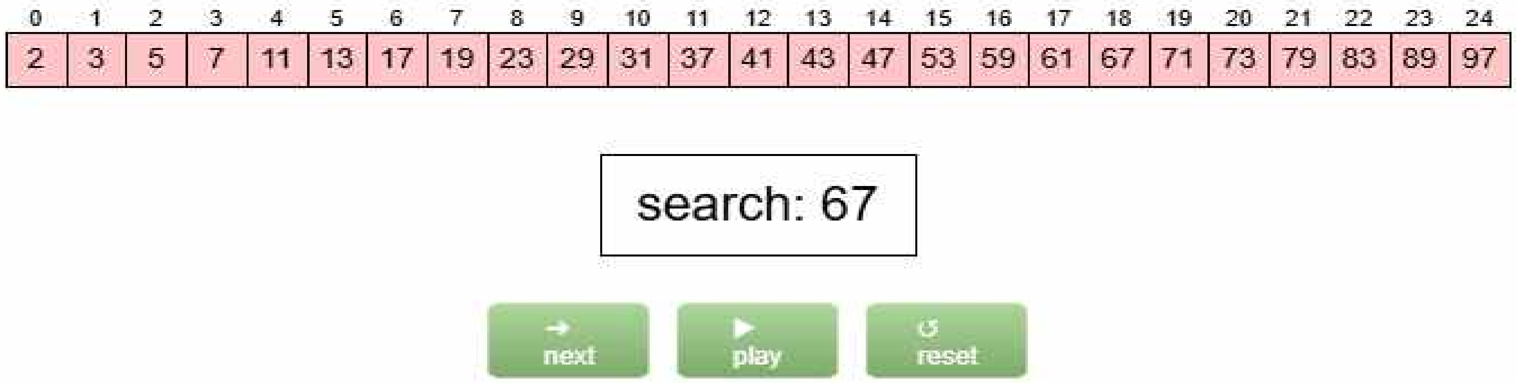
Primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97];

Suppose we want to know whether the number 67 is prime.
If 67 is in the array, then it's prime.

We might also want to know how many primes are smaller than 67.

If we find the position of the number 67 in the array, we can

Once we know that the prime number 67 is at index 18, we can identify that it is a prime. We can also quickly identify that there are 18 elements which come before 67 in the array, meaning that there are 18 prime numbers smaller than 67.



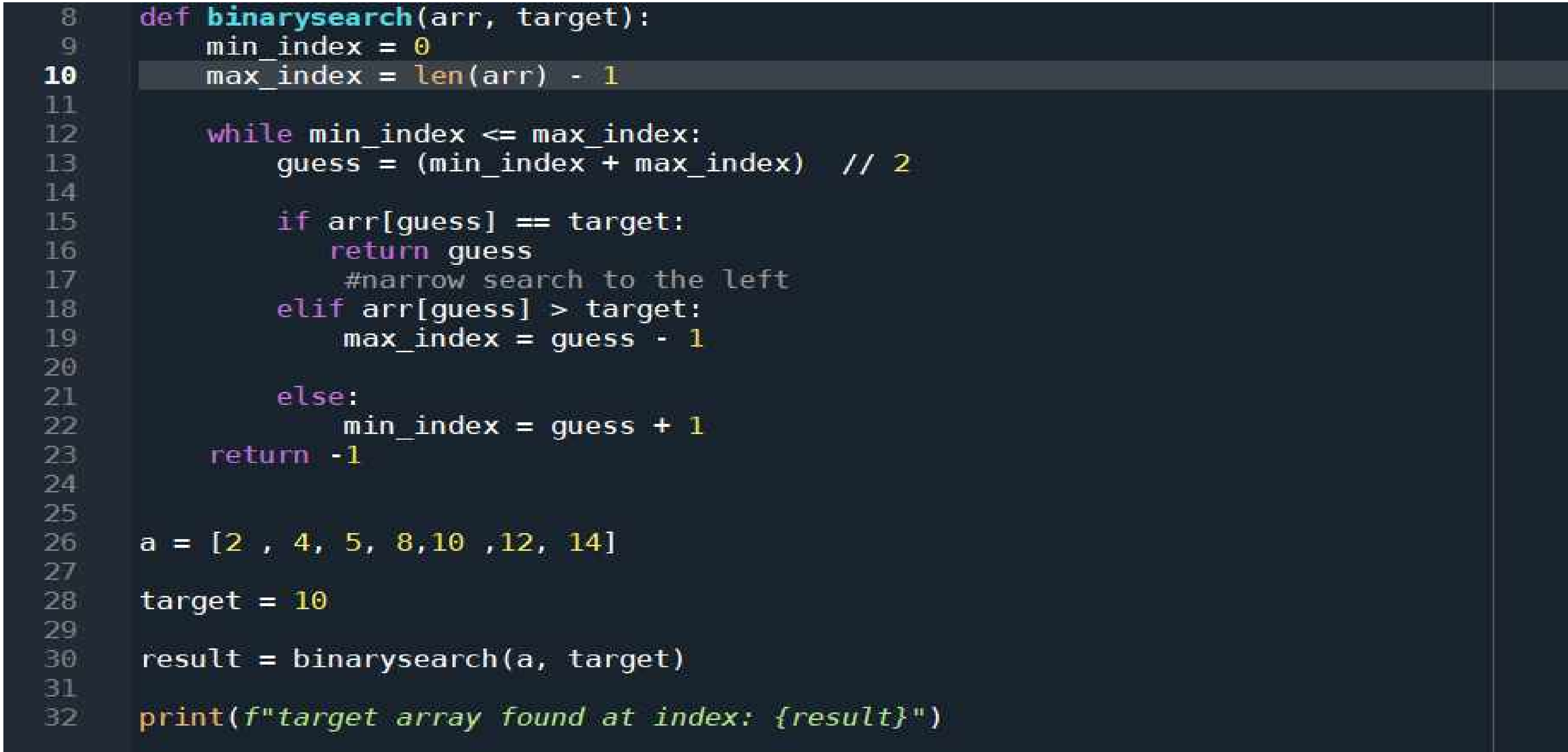
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97

search: 67

Pseudocode of binary search

```
1. Let min = 0 and max = n-1.
2. Compute guess as the average of max and min, rounded down (so that it is an integer).
3. If array[guess] equals target, then stop. You found it! Return guess.
4. If the guess was too low, that is, array[guess] < target, then set min = guess + 1.
5. Otherwise, the guess was too high. Set max = guess - 1.
6. Go back to step 2
```

Pseudocode implementation

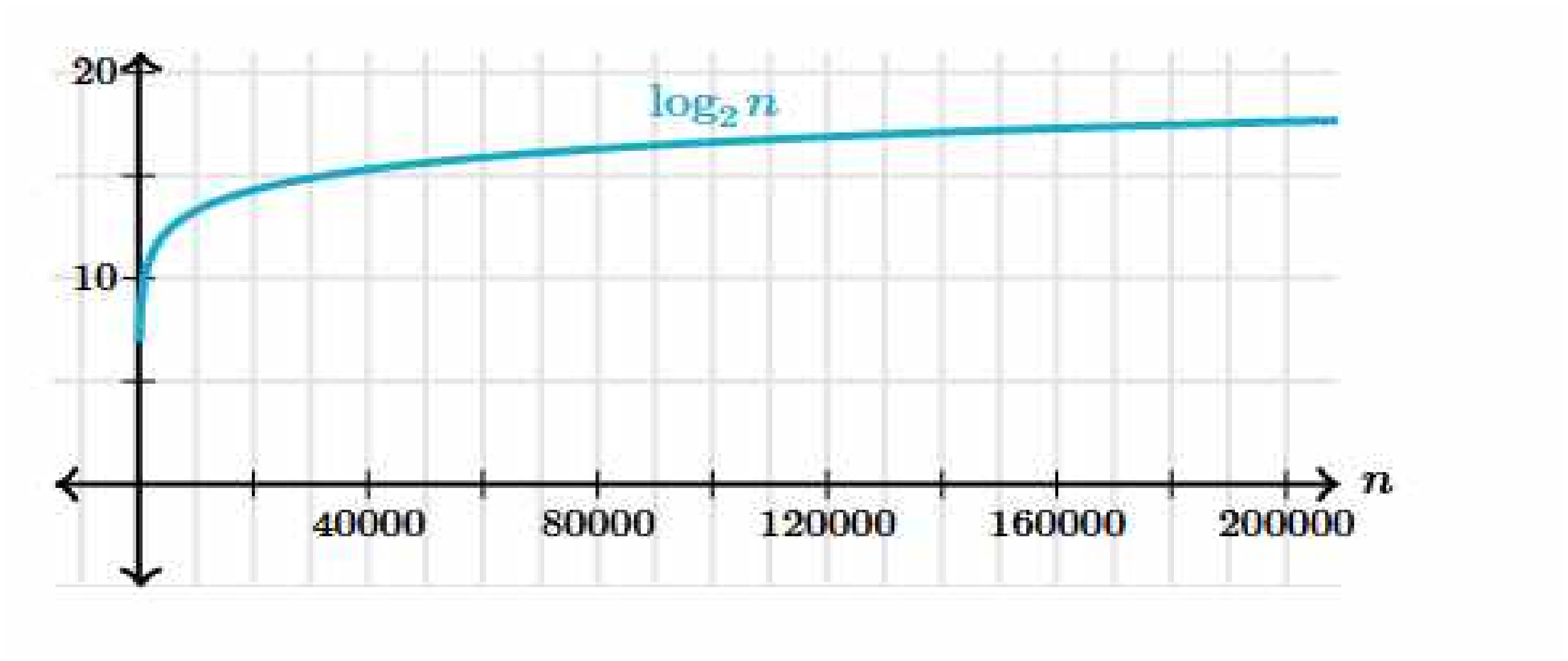


```python
def binarysearch(arr, target):
    min_index = 0
    max_index = len(arr) - 1

    while min_index <= max_index:
        guess = (min_index + max_index)  // 2

        if arr[guess] == target:
            return guess
             #narrow search to the left
        elif arr[guess] > target:
             max_index = guess - 1

        else:
             min_index = guess + 1
    return -1


a = [2 , 4, 5, 8,10 ,12, 14]

target = 10

result = binarysearch(a, target)

print(f"target array found at index: {result}")
```

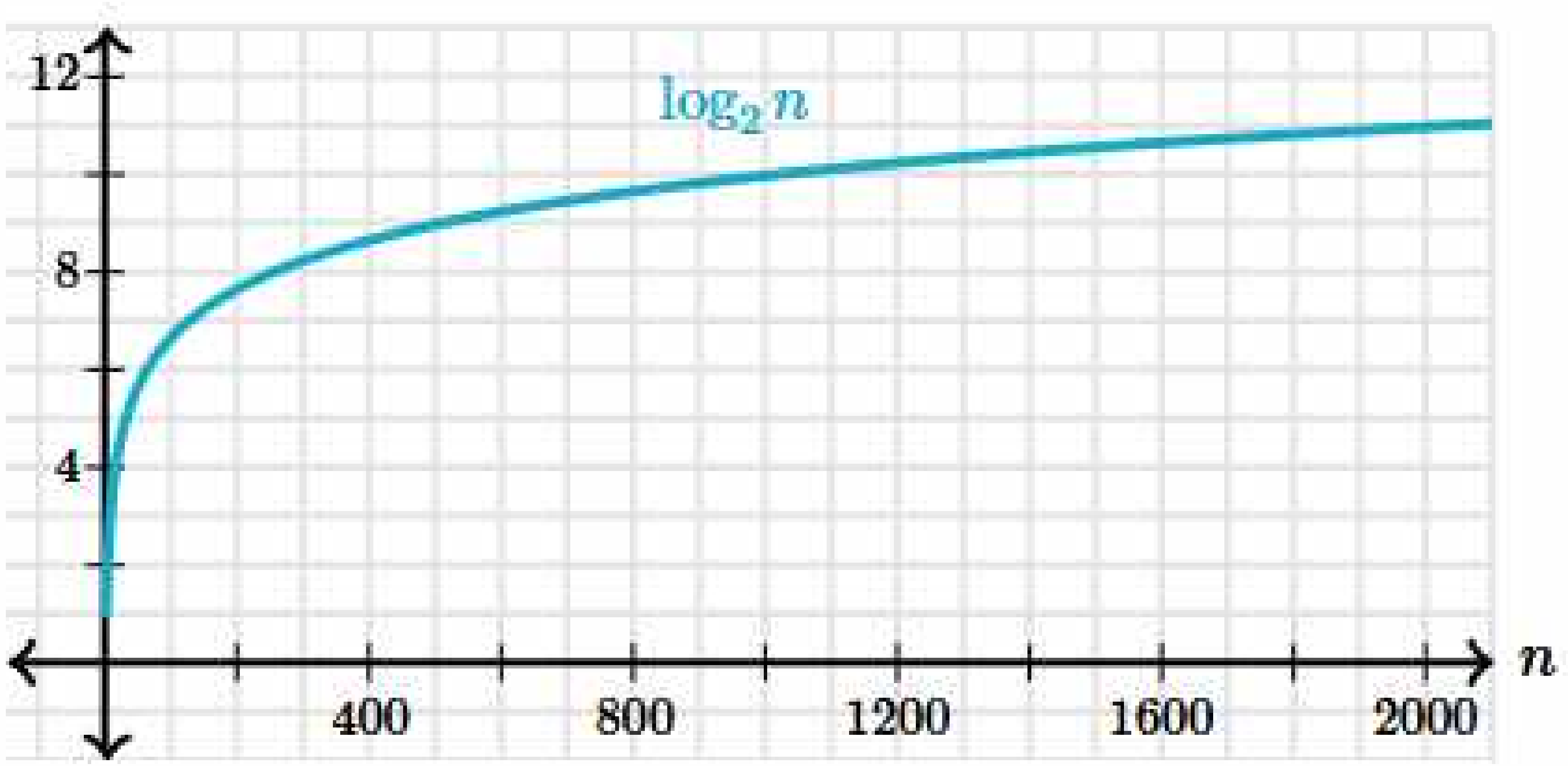
Running time of binary search

N	Log_2 n
1	0
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024	10

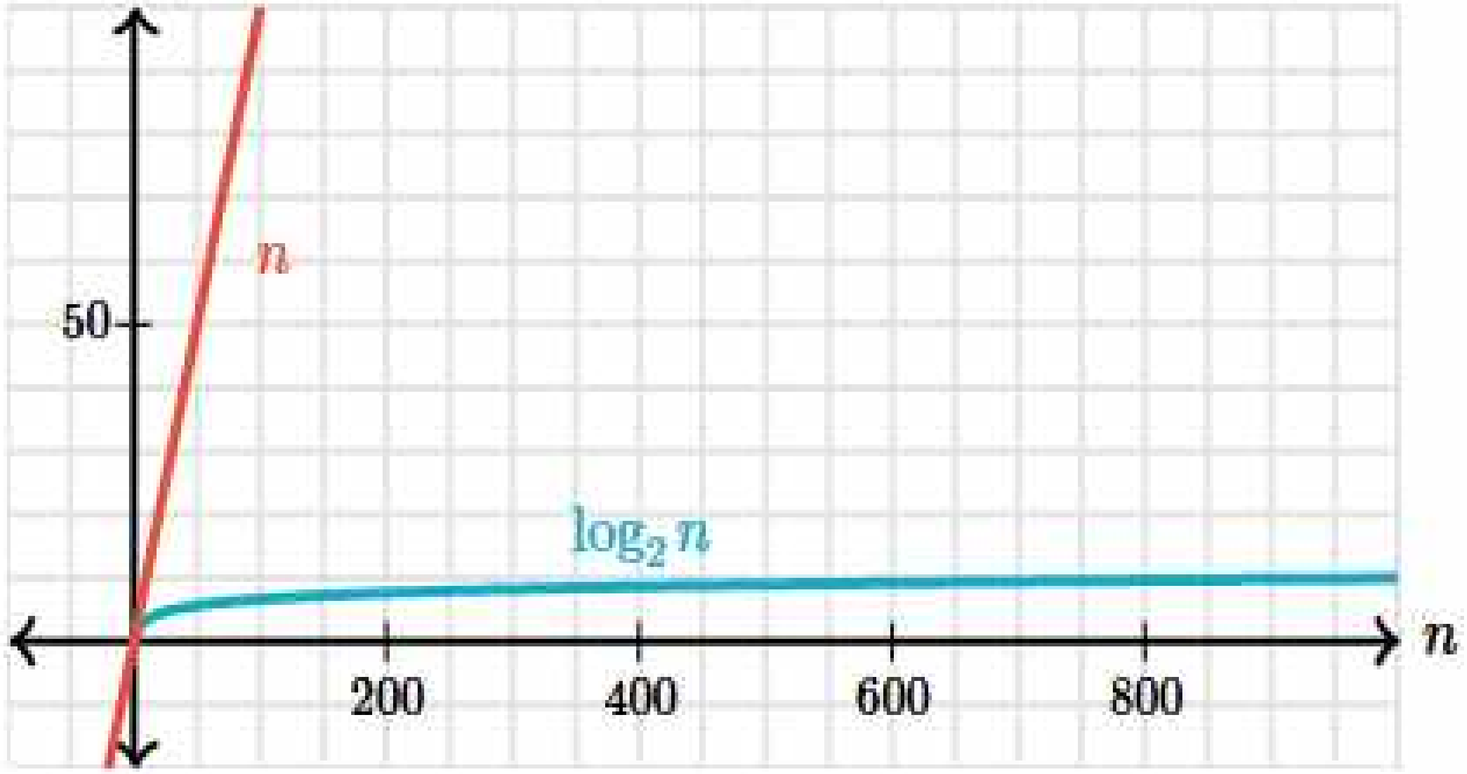
We can view this same table as a graph:



Zooming in on smaller values of n:



Compare N vs log _2 N below:



Thanks for your attention

Any questions are appreciated