Assignment 2

Manan Patel

September 2024

1. The average cost of the algorithm is known to be O(lg n). Derive this conclusion by yourself, step by step, and explicitly explain the assumptions on probability behind the conclusion.

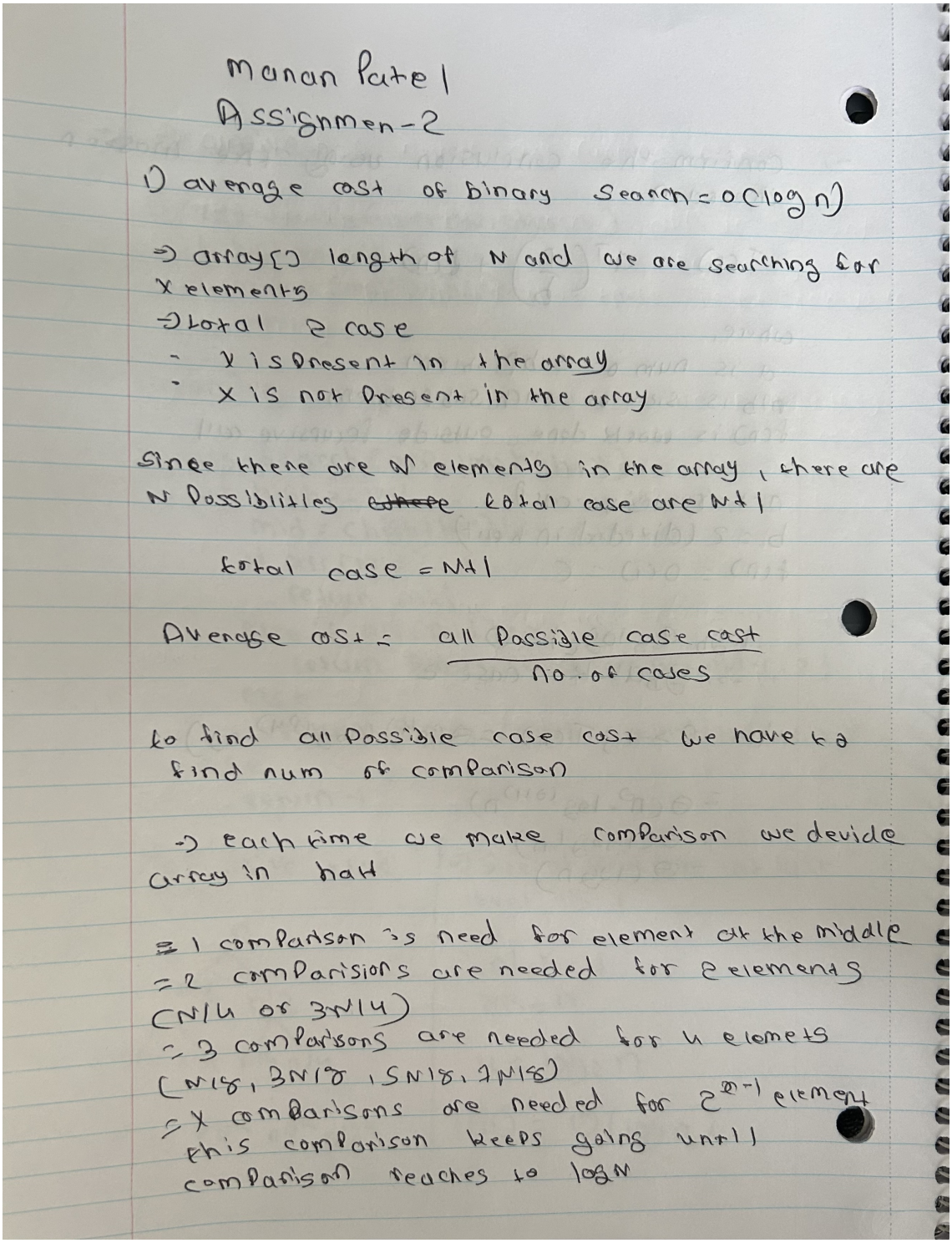

Figure 1: Answer of Task 1

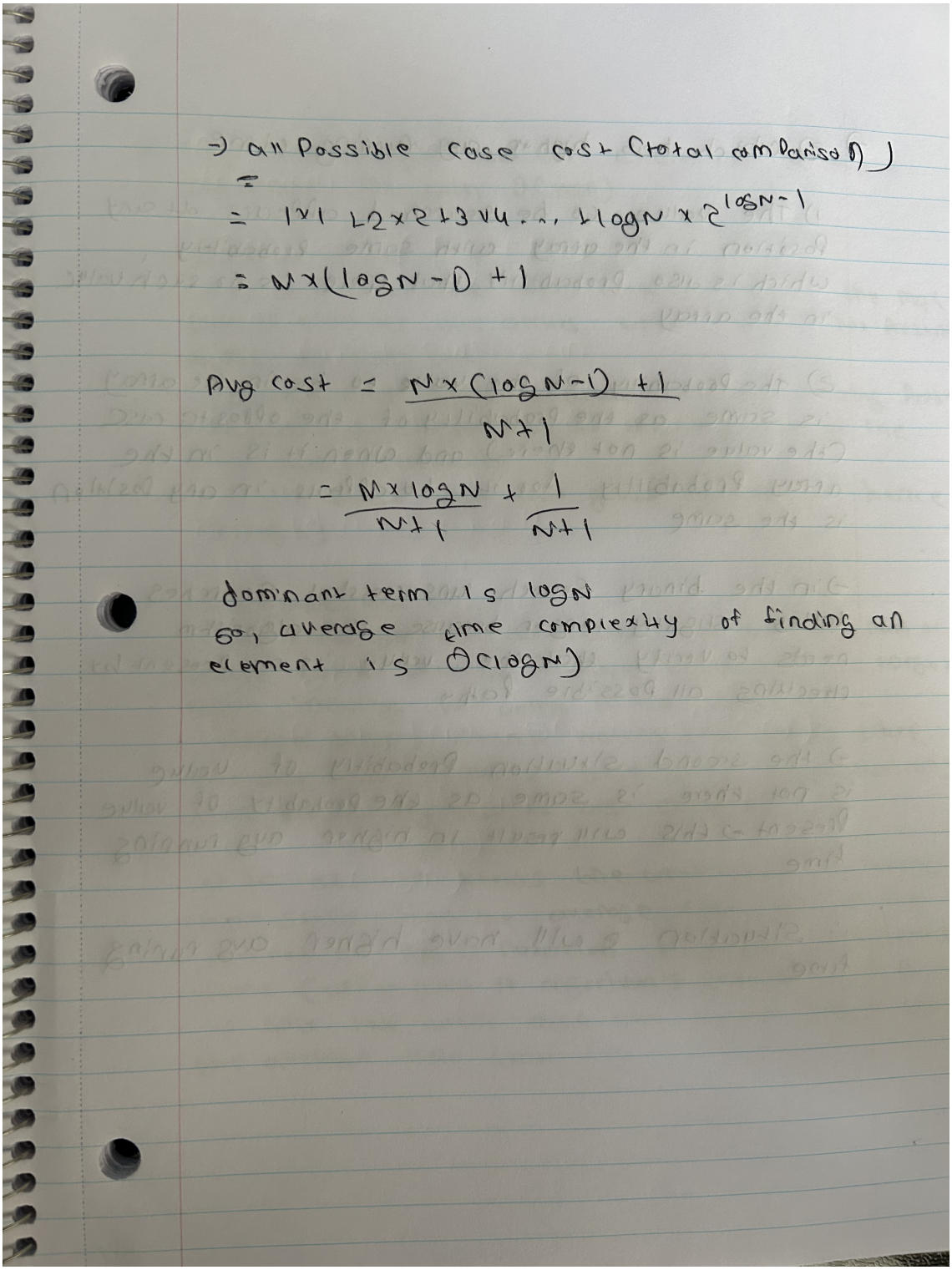

Figure 2: Answer of Task 1

2. Decide whether the algorithm has the same average running time in the following two situations, and if not, which one is higher. Justify your conclusion (in English).

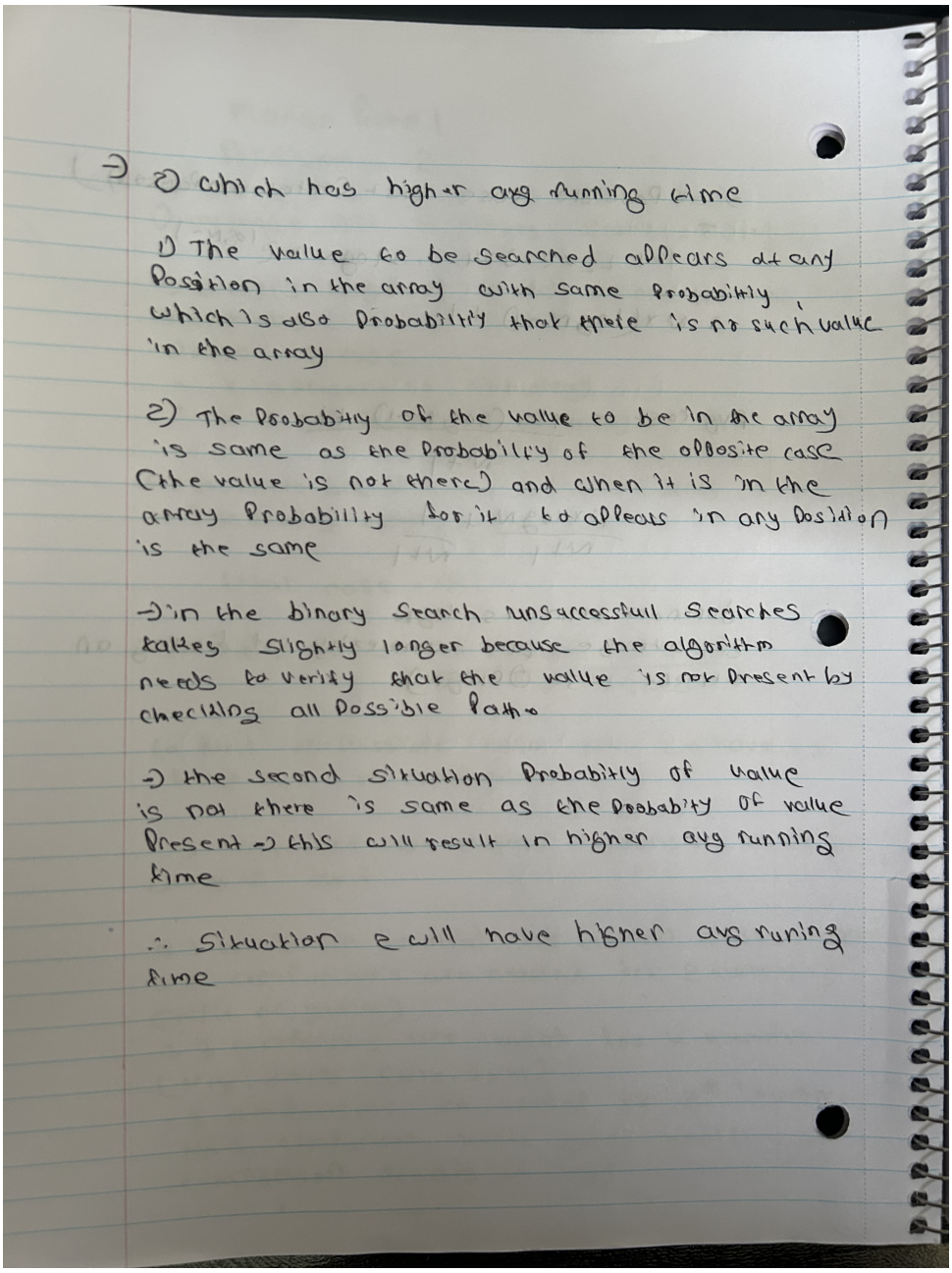

Figure 3: Answer of Task 2

3. Explain (in English) why the average cost and the worst cost of this algorithm are both O(lg n) – shouldn't the worst cost be higher than the average cost?

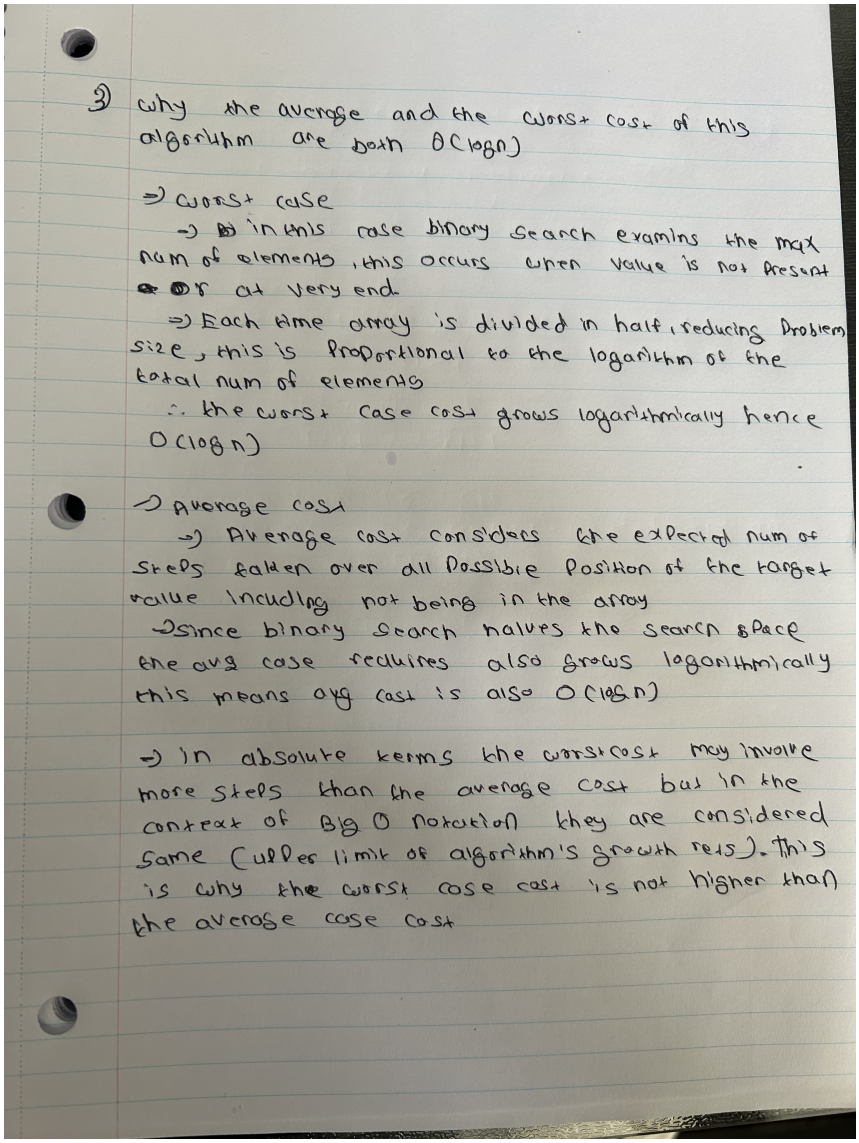

Figure 4: Answer of Task 3

4. Write a program to estimate the average cost of the algorithm using random numbers in arrays of various sizes. Discuss the results and compare it with the above analysis. The requirements are similar to the programming part of Assignment 1.

Testing Results

The binary search algorithm was tested on arrays of various sizes, ranging from 2,000 to 20,000 elements, with 100 trials per array size.Here's the breakdown of the results:

Array Size	Average Comparisons
2000	21.88
4000	23.96
6000	25.34
8000	25.96
10000	26.86
12000	27.34
14000	27.62
16000	27.96
18000	28.34
20000	28.76

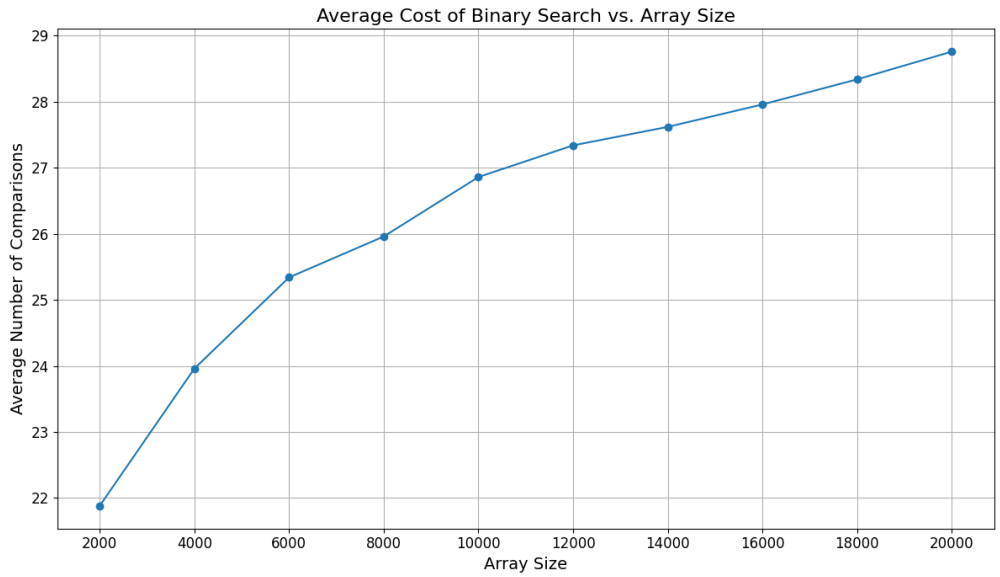

Figure 5: Average Cost of Binary Search for Different Array Sizes. Each point represents the average number of comparisons needed for 100 binary searches on an array of the specified size. The experiments included a mix of successful and unsuccessful searches, with targets randomly selected from within and outside the array.

The average number of comparisons was calculated using the formula:

$$\text{Average Comparisons} = \frac{\text{Total Comparisons}}{\text{Number of Tests}}$$

As the array size increased, the average number of comparisons also increased. This is expected because larger arrays require more elements to be checked during the search process.

The results demonstrate the expected logarithmic growth of the number of comparisons as the array size increases. This is due to the nature of binary search, which systematically halves the search range. For example, the average number of comparisons for an array size of 2,000 was around 21.88, while for 20,000 elements, it was around 28.76. This slow increase is characteristic of $O(\log n)$ complexity.

Moreover, the diminishing difference in average comparisons between larger array sizes (e.g., between 18,000 and 20,000) highlights that the growth rate tapers off, further emphasizing the logarithmic nature of binary search. This matches my earlier theoretical analysis, which concluded that the average cost of the binary search algorithm would be $O(\log n)$.