- Consider the worst-case scenario for the algorithm.
- Identify any loops or recursive functions in the algorithm.
- Use big O notation to express the time complexity.

## **Exploring Time Complexity**

When analyzing the time complexity of an algorithm, the first step is to understand the purpose of the algorithm and the inputs it requires. It's important to consider the worst-case scenario for the algorithm, as well as any loops or recursive functions it may contain. Using big O notation can help to express the time complexity in a clear and concise way.

By mastering the tricks outlined above, you can gain a better understanding of time complexity and become more confident in analyzing and solving related problems.

If you want to determine time complexity, the first technique is to drop the non-dominant terms. This means you drop whatever non-dominant terms you have. For example, if you are given a faller Notes

Drop the non-dominant terms

This will help simplify the calculation of tire complexity.

I am furrently writing a simple for loop in the C programming language. The loop will run n times, and is written as:

```
for (i = 0; i < n; i++)
```

Now, this loop is performing a certain task. To understand what it does, follow these steps:

- 1. Set i equal to 0.
- 2. Check if i is less than n.
- 3. If i is less than n, perform the task associated with the loop.
- 4. Add 1 to i.
- 5. Return to step 2.