We assume that changing the platform will change the execution time of the program by a constant factor, and that we can therefore ignore platform differences in conformance with the asymptotically equivalent principle described earlier.

To place this discussion in context, we briefly discuss the **Sequential Search** algorithm, presented later in Chapter 5. **Sequential Search** examines a list of $n \geq 1$ distinct elements, one at a time, until a desired value, v, is found. For now, assume that:

- There are n distinct elements in the list
- The list contains the desired value v
- Each element in the list is equally likely to be the desired value v

To understand the performance of **Sequential Search**, we must know how many elements it examines "on average." Since v is known to be in the list and each element is equally likely to be v, the average number of examined elements, $E(n)$, is the sum of the number of elements examined for each of the n values divided by n. Mathematically:

$$E(n) = \frac{1}{n}\sum_{i=1}^{n} i = \frac{n(n+1)}{2n} = \frac{1}{2}n + \frac{1}{2}$$

Thus, **Sequential Search** examines about half of the elements in a list of n distinct elements subject to these assumptions. If the number of elements in the list doubles, then **Sequential Search** should examine about twice as many elements; the expected number of probes is a *linear* function of n. That is, the expected number of probes is "about" $c*n$ for some constant c; here, $c = 0.5$. A fundamental fact of performance analysis is that the constant c is unimportant in the long run, because the most important cost factor is the size of the problem instance, n. As n gets larger and larger, the error in claiming that:

$$\frac{1}{2}n \approx \frac{1}{2}n + \frac{1}{2}$$

becomes less significant. In fact, the ratio between the two sides of this approximation approaches 1. That is:

$$\lim_{n \to \infty} \frac{\left(\frac{1}{2}n\right)}{\left(\frac{1}{2}n + \frac{1}{2}\right)} = 1$$

although the error in the estimation is significant for small values of n. In this context, we say the rate of growth of the expected number of elements that **Sequential Search** examines is linear. That is, we ignore the constant multiplier and are concerned only when the size of a problem instance is large.