Worth: 3% **Due:** By 12 noon on Tuesday 3 April.

---

1. **Intuition: Depending on the value of L[0], we either execute a for-loop that has n^2 iterations or a for-loop that has n iterations. Since there is a constant number of steps in either iteration, the total number of steps will not be more than a constant times n^2. For this reason, our intutition tells us that the worst-case running time is $\mathcal{O}(n^2)$.**

   To determine a lower-bound on the worst-case running time, we need to think about what sort of input produces a worst-case time. The worst-case time is triggered when L[0] is even, and then n^2 iterations of a for-loop are executed. Since each iteration will take at least one-step, our intuition tells us that the worst-case running time is $\Omega(n^2)$.

   Altogether, then, the worst-case running time is $\Theta(n^2)$.

   To prove this conclusion in a detailed way, we need to put the above arguments in a formal proof that follows the usual structure.

   Let I be the set of possible inputs to the algorithm. That is, I is the set of nonempty lists of numbers.

   Let $c_0 = 3$ and $B_0 = 1$.
   Then $c_0 \in \mathbb{R}^+$ and $B_0 \in \mathbb{N}$.
   Assume L $\in I$ is an arbitrary list of length $n \geqslant B_0$
     Then we need to consider the single if-statement plus the time taken in either the if- or else- clauses.
     Then L[0] is either even or odd.
     Case 1: Assume L[0] is even.
       Then the algorithm executes 2 statements for each iteration of a for-loop (lines 2,3), and loops for (at most) n^2 iterations.
       Then at most $2n^2$ statements are executed.
     Case 2: Assume L[0] is odd.
       Then the algorithm executes 2 statements for each iteration of a for-loop (lines 5,6), and loops for (at most) n iterations.
       Then at most $2n$ statements are executed.
       Then at most $2n^2$ statements are executed.
     Then, in either case, at most $2n^2$ statements are executed.
     Then, in total, the algorithm requires at most $2n^2 + 1$ statements to be executed.
     Then $t(L) \leqslant 2n^2 + 1$.
     Then $t(L) \leqslant 3n^2$.
     Then $t(L) \leqslant c_0 n^2$.
   Then $\forall L \in I, \text{size(L)} \geqslant B_0 \Rightarrow t(\text{L}) \leqslant c_0 n^2 \quad \#\ \text{size(L)} = n$
   Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall L \in I, \text{size(L)} \geqslant B \Rightarrow t(\text{L}) \leqslant cn^2$
   Then $T(n) \in \mathcal{O}(n^2)$.

   Let $c_0 = 1$ and $B_0 = 1$.
   Then $c_0 \in \mathbb{R}^+$ and $B_0 \in \mathbb{N}$.
   Assume $n \in \mathbb{N}$ and $n \geqslant B_0$
     Let $\text{L}_0 = [2, 2, 3, \ldots, n]$.
     Then $\text{L}_0 \in I$ and $\text{size}(\text{L}_0) = n$.
     Then $\text{L}_0[0]$ is even.
     Then n^2 iterations of at least one statement are executed.
     Then, including the if-statement, at least $n^2 + 1$ statements executed.

Then $t(L_0) \geqslant n^2 + 1$.
Then $t(L_0) \geqslant n^2$.
Then $t(L_0) \geqslant c_0 n^2$.
Then $\text{size}(L_0) = n \wedge t(L_0) \geqslant c_0 n^2$
Then $\exists L \in I, \text{size}(L) = n \wedge t(L_0) \geqslant c_0 n^2$

Then $\forall n \in \mathbb{N}, n \geqslant B_0 \Rightarrow \exists L \in I, \text{size}(L) = n \wedge t(L_0) \geqslant c_0 n^2$
Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geqslant B_0 \Rightarrow \exists L \in I, \text{size}(L) = n \wedge t(L_0) \geqslant c_0 n^2$
Then $T(n) \in \Omega(n^2)$.

Then $T(n) \in \Theta(n^2)$.

Here, as is usual, $T(n)$ is the worst-case running time for the algorithm on a list of length n.

2. Intuition: The number of iterations of the loop on any input L of length n will depend on the value of the variable index. And the value of index depends on the value of variable step. The variable step increases by 1 on each iteration of the loop.

The variable step takes on the values:

$$\text{step} = 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow \ldots$$

The variable index takes on the values:

$$\text{index} = 0 \rightarrow 0+1 \rightarrow 0+1+2 \rightarrow 0+1+2+3 \rightarrow \ldots \rightarrow 0+1+2+3+\ldots+k \rightarrow \ldots$$

The value of index after the k^{th} iteration of the loop is $0+1+2+3+\ldots+k = \sum_{j=0}^{j=k} j = k(k+1)/2$.

The number of iterations of the loop will be m, where m is such that $m(m+1)/2 \geqslant n$ (to get out of the loop) while $(m-1)((m-1)+1)/2 < n$. (To have a m^{th} iteration of the loop, we need $(m-1)((m-1)+1)/2 < n$ or $(m-1)m/2 < n$.)

So we have $m(m-1) < 2n$ and $m(m+1) \geqslant 2n$. Hence, there will be $m \approx \lceil\sqrt{2n}\rceil$ iterations. (There won't be more than $\lceil\sqrt{2n}\rceil$ iterations. There won't be fewer than $\lfloor\sqrt{2n}\rfloor$ iterations.)

We can justify this last conclusion more rigorously. Consider the expression $m(m-1)$. We want $m(m-1) < 2n$. Suppose $m = \lceil\sqrt{2n}\rceil + 1$. Then, considering $m(m-1)$, we have

$$\begin{aligned}
(\lceil\sqrt{2n}\rceil + 1)((\lceil\sqrt{2n}\rceil + 1) - 1) &= (\lceil\sqrt{2n}\rceil + 1)(\lceil\sqrt{2n}\rceil) \\
&= (\lceil\sqrt{2n}\rceil)^2 + \lceil\sqrt{2n}\rceil \\
&\geqslant (\sqrt{2n})^2 + \sqrt{2n} \\
&= 2n + \sqrt{2n} \\
&> 2n \qquad \text{since } n > 0
\end{aligned}$$

Hence, $m \leqslant \lceil\sqrt{2n}\rceil$, since otherwise $m(m-1) \not< 2n$.

Similarly, we want $m(m+1) \geqslant 2n$. Suppose $m = \lfloor\sqrt{2n}\rfloor - 1$. Then

$$\begin{aligned}
(\lfloor\sqrt{2n}\rfloor - 1)((\lfloor\sqrt{2n}\rfloor - 1) + 1) &= (\lfloor\sqrt{2n}\rfloor - 1)(\lfloor\sqrt{2n}\rfloor) \\
&\leqslant (\sqrt{2n} - 1)(\sqrt{2n}) \\
&= (\sqrt{2n})^2 - \sqrt{2n} \\
&= 2n - \sqrt{2n} \\
&< 2n \qquad \text{since } n > 0
\end{aligned}$$

Hence, $m \geqslant \lfloor\sqrt{2n}\rfloor$, since otherwise $m(m+1) \not\geqslant 2n$.

Since the number of steps per iteration is constant, and the number of steps performed only depends on the length of the list L, we have that the worst-case runtime of the algorithm is $\Theta(\sqrt{n})$, where $n = \text{len}(\text{L})$.

To prove this conclusion in a detailed way, we need to put the above arguments in a formal proof that follows the usual structure.

Let I be the set of possible inputs to the algorithm. That is, I is the set of nonempty lists of numbers.

> **Let $c_0 = 4\sqrt{2} + 7$ and $B_0 = 1$.**
> **Then $c_0 \in \mathbb{R}^+$ and $B_0 \in \mathbb{N}$.**
> **Assume $\text{L} \in I$ is an arbitrary list of length $n \geqslant B_0$**
>> **Then the algorithm executes 4 statements for each iteration of the loop (lines 4,5,6,7), and loops for at most $\lceil\sqrt{2n}\rceil$ iterations.**
>> **In addition, the algorithm has 2 initialization statements and the last evaluation of the loop condition.**
>> **Then,**
>> $$\begin{aligned} t(\text{L}) &\leqslant 4\lceil\sqrt{2n}\rceil + 3 \\ &\leqslant 4(\sqrt{2n} + 1) + 3 \quad \text{\# add 1 to remove ceiling} \\ &= 4\sqrt{2}\sqrt{n} + 7 \\ &\leqslant 4\sqrt{2}\sqrt{n} + 7\sqrt{n} \\ &= (4\sqrt{2} + 7)\sqrt{n} \\ &= c_0\sqrt{n} \end{aligned}$$
>
> **Then $\forall L \in I, \text{size}(\text{L}) \geqslant B_0 \Rightarrow t(\text{L}) \leqslant c_0\sqrt{n}$ # $\text{size}(\text{L}) = n$**
> **Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall L \in I, \text{size}(\text{L}) \geqslant B \Rightarrow t(\text{L}) \leqslant c\sqrt{n}$**
> **Then $T(n) \in \mathcal{O}(\sqrt{n})$.**
>
> **Let $c_0 = \sqrt{2}$ and $B_0 = 1$.**
> **Then $c_0 \in \mathbb{R}^+$ and $B_0 \in \mathbb{N}$.**
> **Assume $n \in \mathbb{N}$ and $n \geqslant B_0$**
>> **Let $\text{L}_0 = [1, 2, 3, \ldots, n]$.**
>> **Then $\text{L}_0 \in I$ and $\text{size}(\text{L}_0) = n$.**
>> **Then the algorithm executes 2 steps followed by at least $\lfloor\sqrt{2n}\rfloor$ iterations of at least one statement.**
>> **Then**
>> $$\begin{aligned} t(\text{L}_0) &\geqslant \lfloor\sqrt{2n}\rfloor + 2 \\ &> \sqrt{2n} \\ &= \sqrt{2}\sqrt{n} \\ &= c_0\sqrt{n} \end{aligned}$$
>> **Then $t(\text{L}_0) \geqslant c_0\sqrt{n}$**
>> **Then $\text{size}(\text{L}_0) = n \wedge t(\text{L}_0) \geqslant c_0\sqrt{n}$**
>> **Then $\exists \text{L} \in I, \text{size}(\text{L}) = n \wedge t(\text{L}_0) \geqslant c_0\sqrt{n}$**
>
> **Then $\forall n \in \mathbb{N}, n \geqslant B_0 \Rightarrow \exists \text{L} \in I, \text{size}(\text{L}) = n \wedge t(\text{L}_0) \geqslant c_0\sqrt{n}$**
> **Then $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geqslant B_0 \Rightarrow \exists \text{L} \in I, \text{size}(\text{L}) = n \wedge t(\text{L}_0) \geqslant c_0\sqrt{n}$**
> **Then $T(n) \in \Omega(\sqrt{n})$.**
>
> **Then $T(n) \in \Theta(\sqrt{n})$.**

Here, as is usual, $T(n)$ is the worst-case running time for the algorithm on a list of length n.