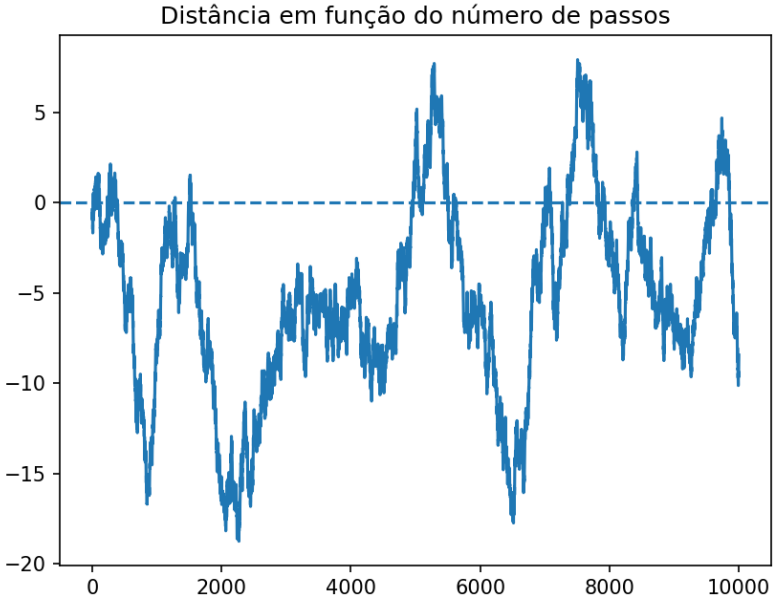
```python
"""Atividade 8: Caminhadas Aleatórias."""
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm

def random_walk(steps: int, dimensions: int):
    current_pos = np.zeros(shape=(steps, dimensions))
    for i in range(steps):
        direction = np.random.uniform(low=-0.5, high=0.5, size=dimensions)
        current_pos[i] = direction + current_pos[i - 1]
    return current_pos

positions = random_walk(10_000, 1)
plt.title("Distância em função do número de passos")
plt.plot(positions)
plt.axhline(y=0, linestyle="dashed")
```

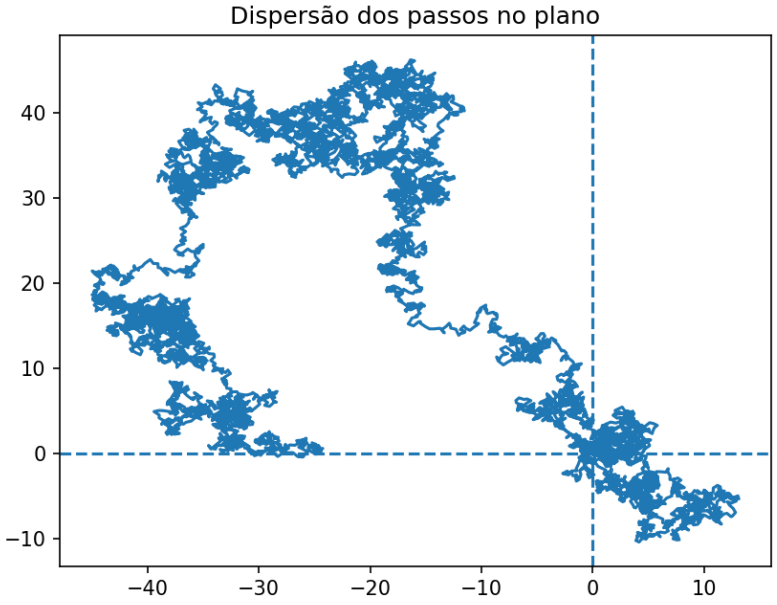
```
Installed gtk3 event loop hook.
```




```
<matplotlib.lines.Line2D at 0x7fc86f7eeb10>
```

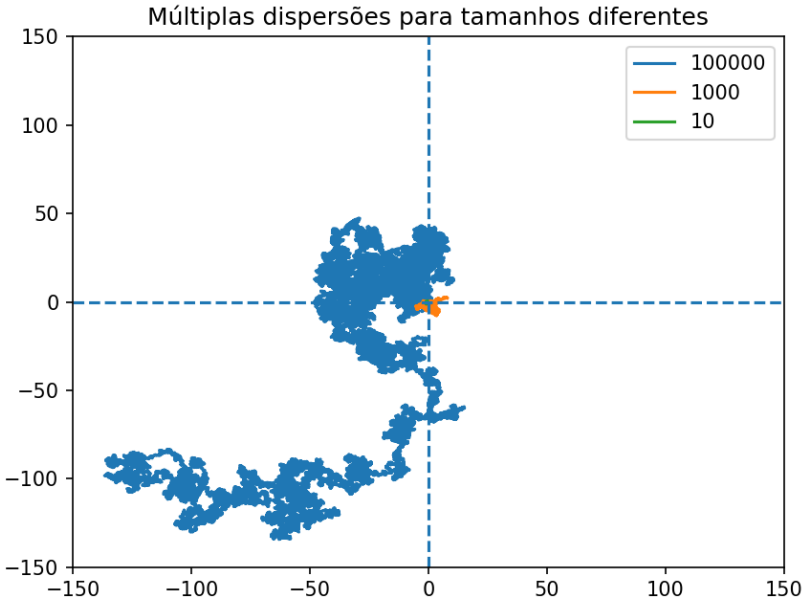
```python
positions = random_walk(10_000, 2)
plt.title("Dispersão dos passos no plano")
plt.plot(positions[:, 0], positions[:, 1])
```

```
plt.axhline(y=0, linestyle="dashed")
plt.axvline(x=0, linestyle="dashed")
```




```
<matplotlib.lines.Line2D at 0x7fc89edadb10>
```

```
for j in [100_000, 1_000, 10]:
    z = int(round(j, 0))
    positions = random_walk(z, 2)
    plt.plot(positions[:, 0], positions[:, 1], label=z)
plt.axhline(y=0, linestyle="dashed")
plt.axvline(x=0, linestyle="dashed")
plt.xlim([-150, 150])
plt.ylim([-150, 150])
plt.legend()
plt.title("Múltiplas dispersões para tamanhos diferentes")
```

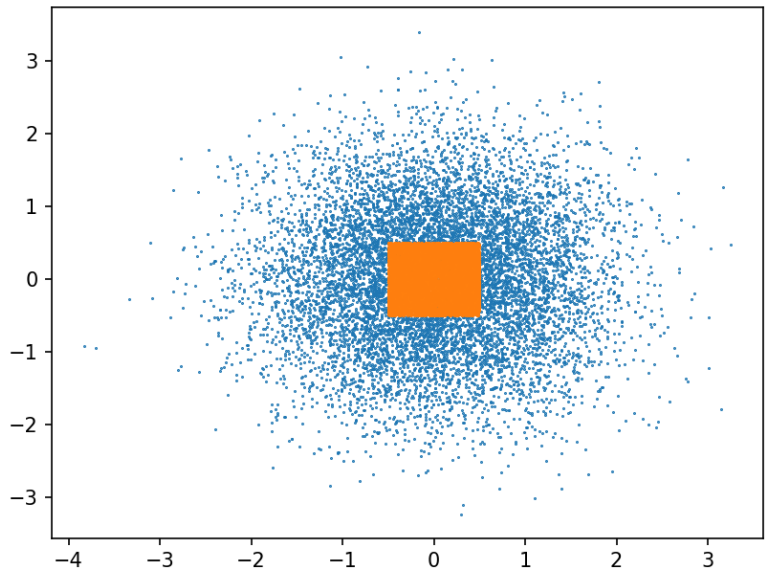



```
Text(0.5, 1.0, 'Múltiplas dispersões para tamanhos diferentes')
```

B

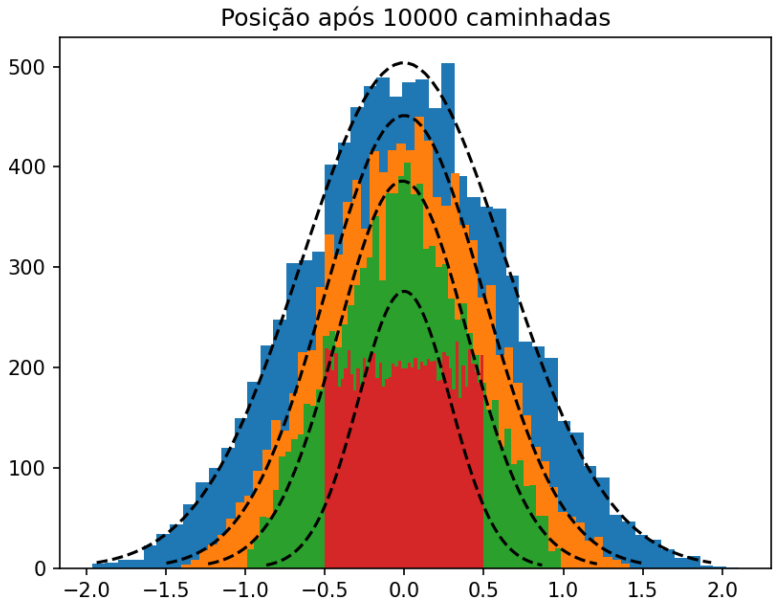
Não, a distância final não aumenta por cerca de 10 vezes. Parece não existir um padrão mutio claro.

```
def final_walk(w: int):
    for i in [10, 1]:
        last = np.zeros(shape=(w, 2))
        for j in range(w):
            last_step = random_walk(i, 2)
            last[j] = np.array([last_step[-1, 0], last_step[-1, 1]])
        plt.scatter(x=last[:, 0], y=last[:, 1], s=0.2)

final_walk(10_000)
```




```
def estimate(steps: int, w: int):
    walks = np.zeros(shape=(w, 1))
    for j in range(w):
        walk = random_walk(steps, 1)
        walks[j] = walk[-1, 0]

    # No enunciado fala pra usar isso aqui
    # rms = np.sqrt(np.mean(walks**2))

    bins = 50
    plt.hist(walks[:, 0], bins=bins)

    # Faz a "escala" do gráfico
    bin_width = (walks.max() - walks.min()) / bins
    hist_area = len(walks[:, 0]) * bin_width

    mu, std = norm.fit(walks)

    x = np.linspace(mu - 3 * std, mu + 3 * std, 100)
    plt.plot(x, norm.pdf(x, mu, std) * hist_area, linestyle="dashed", color="black")

N = 10_000
```

```
for i in [5, 3, 2, 1]:
    estimate(i, N)
    plt.title(f"Posição após {N} caminhadas")
```



C

A Gaussiana se torna uma boa aproximação a partir de $N = 5$, mas deixa de ser ”terrível” logo para $N = 2$.