# GnA Practicum 1: Sorting algorithms

## Plotting and visualizing the data

### Imports

Importing packages for easy data manipulation and visualizations to gain insight into the data.

In [1]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
```

### Reading the data

In [2]:
```
df_qs = pd.read_csv('./quicksort.csv')
df_is = pd.read_csv('./insertionsort.csv')
df_ss = pd.read_csv('./selectionsort.csv')
```

### Showing parts of the data

#### Quicksort

In [3]:
```
df_qs
```

Out[3]:

| | index | amount of compares | size of array |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **1** | 1 | 2 | 2 |
| **2** | 2 | 6 | 3 |
| **3** | 3 | 8 | 4 |
| **4** | 4 | 13 | 5 |
| **...** | ... | ... | ... |
| **995** | 995 | 11251 | 996 |
| **996** | 996 | 11654 | 997 |
| **997** | 997 | 11802 | 998 |
| **998** | 998 | 12050 | 999 |
| **999** | 999 | 13465 | 1000 |

1000 rows × 3 columns

### Insertion sort

In [4]:
```
df_is
```

Out[4]:

| | index | amount of compares | size of array |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **1** | 1 | 0 | 2 |
| **2** | 2 | 2 | 3 |
| **3** | 3 | 3 | 4 |
| **4** | 4 | 2 | 5 |
| **...** | ... | ... | ... |
| **995** | 995 | 249932 | 996 |
| **996** | 996 | 251995 | 997 |
| **997** | 997 | 255344 | 998 |
| **998** | 998 | 242574 | 999 |
| **999** | 999 | 249781 | 1000 |

1000 rows × 3 columns

### Selection sort

In [5]:
```
df_ss
```

Out[5]:

| | index | amount of compares | size of array |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **1** | 1 | 1 | 2 |
| **2** | 2 | 3 | 3 |
| **3** | 3 | 6 | 4 |
| **4** | 4 | 10 | 5 |
| **...** | ... | ... | ... |
| **995** | 995 | 495510 | 996 |
| **996** | 996 | 496506 | 997 |
| **997** | 997 | 497503 | 998 |
| **998** | 998 | 498501 | 999 |
| **999** | 999 | 499500 | 1000 |

1000 rows × 3 columns

## Getting interesting ranges of data

For the table in the report, I used the following sets of data to show partial results.

In [6]:
```
dfs = [df_ss, df_is, df_qs]

arr = []

for df in dfs:
    arr.append((10,int(df[df["size of array"] == 10]["amount of compares"])))
    arr.append((20,int(df[df["size of array"] == 20]["amount of compares"])))
    arr.append((50,int(df[df["size of array"] == 50]["amount of compares"])))
    arr.append((100,int(df[df["size of array"] == 100]["amount of compares"])))
    arr.append((200,int(df[df["size of array"] == 200]["amount of compares"])))
    arr.append((500,int(df[df["size of array"] == 500]["amount of compares"])))
    arr.append((1000,int(df[df["size of array"] == 1000]["amount of compares"])))

for i in arr:
    print(i)
```

```
(10, 45)
(20, 190)
(50, 1225)
(100, 4950)
(200, 19900)
(500, 124750)
(1000, 499500)
(10, 29)
(20, 80)
(50, 575)
(100, 2519)
(200, 10208)
(500, 60828)
(1000, 249781)
(10, 34)
(20, 88)
(50, 314)
(100, 884)
(200, 1964)
```

```
(500, 5411)
(1000, 13465)
```

**The results:**

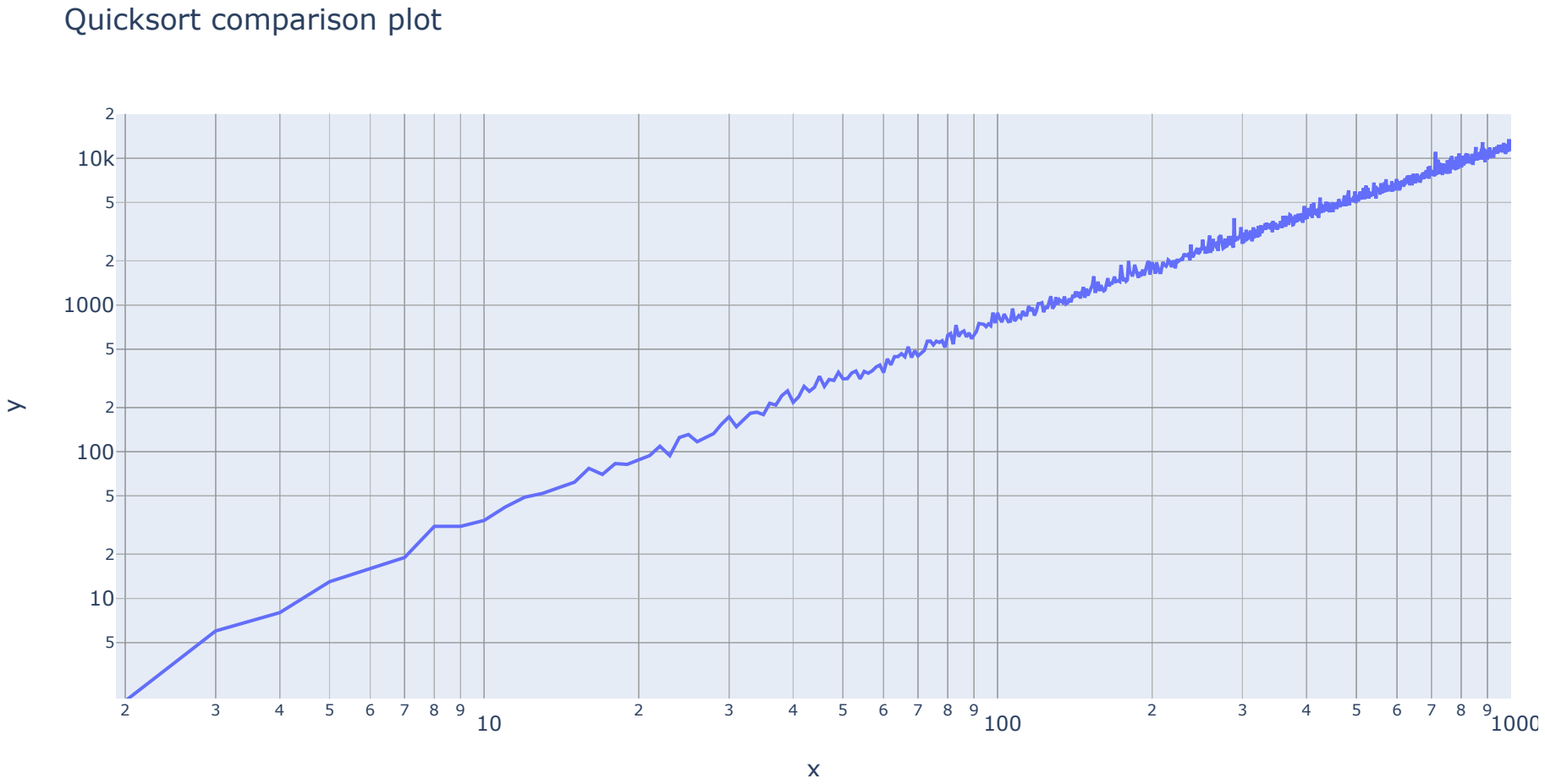
(10, 45)
(20, 190)
(50, 1225)
(100, 4950)
(200, 19900)
(500, 124750)
(1000, 499500)
(10, 29)
(20, 80)
(50, 575)
(100, 2519)
(200, 10208)
(500, 60828)
(1000, 249781)
(10, 34)
(20, 88)
(50, 314)
(100, 884)
(200, 1964)
(500, 5411)
(1000, 13465)

## Plotting on a log-log plot

The plots try to start with values smaller than 1 (0.9, 0.8, ...) and this is why the slopes start with a straight vertical line in the beginning. The Y-axis shows the amount of compares and the X-axis shows the amount of elements in the input array.

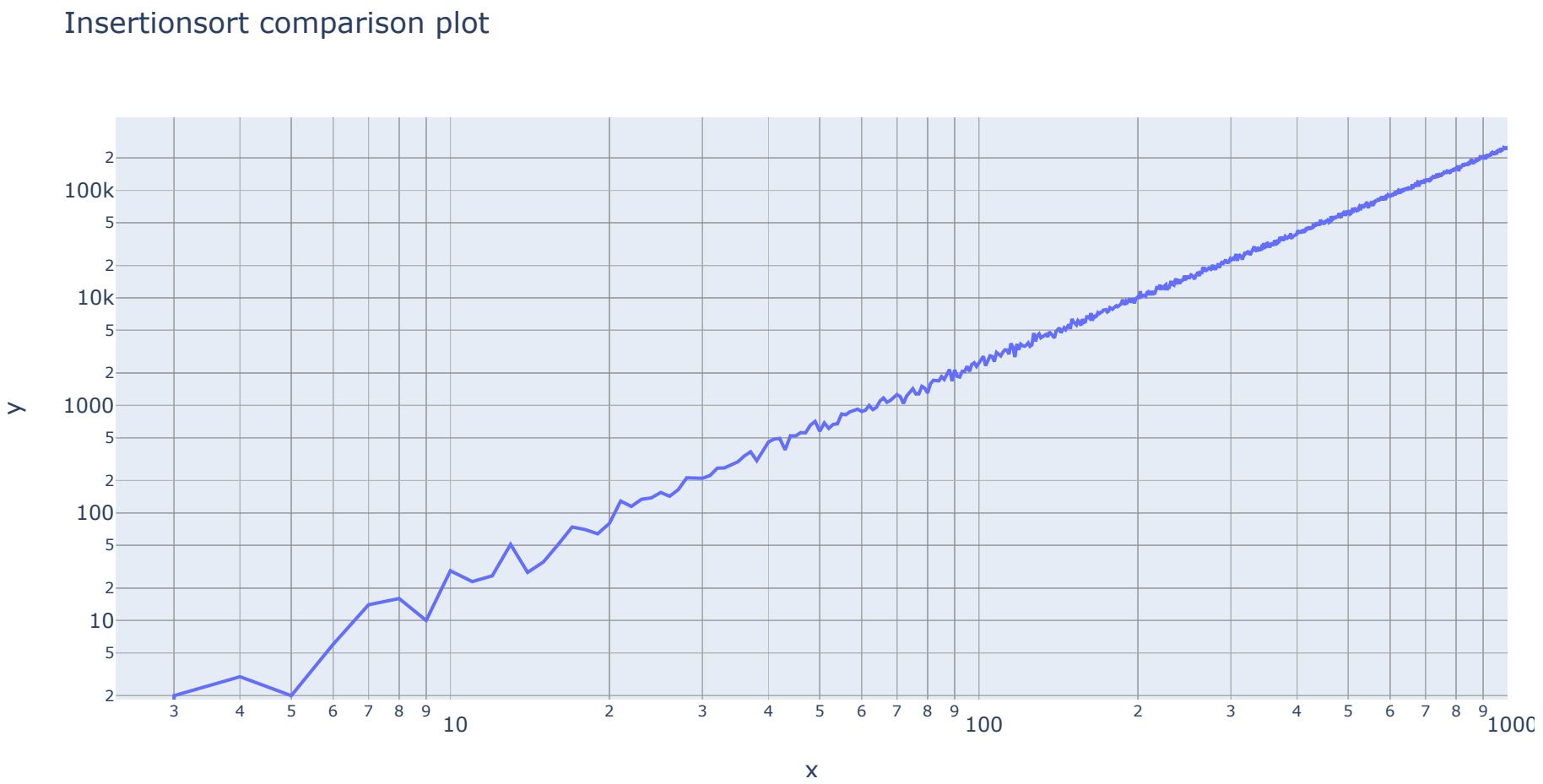
**Quicksort plot:**

In [7]:
```
fig = px.line(df_qs.loc[:, df_qs.columns != "index"], x= df_qs['size of array'], y = df_qs['amount of compares'], title=
fig.show()
```



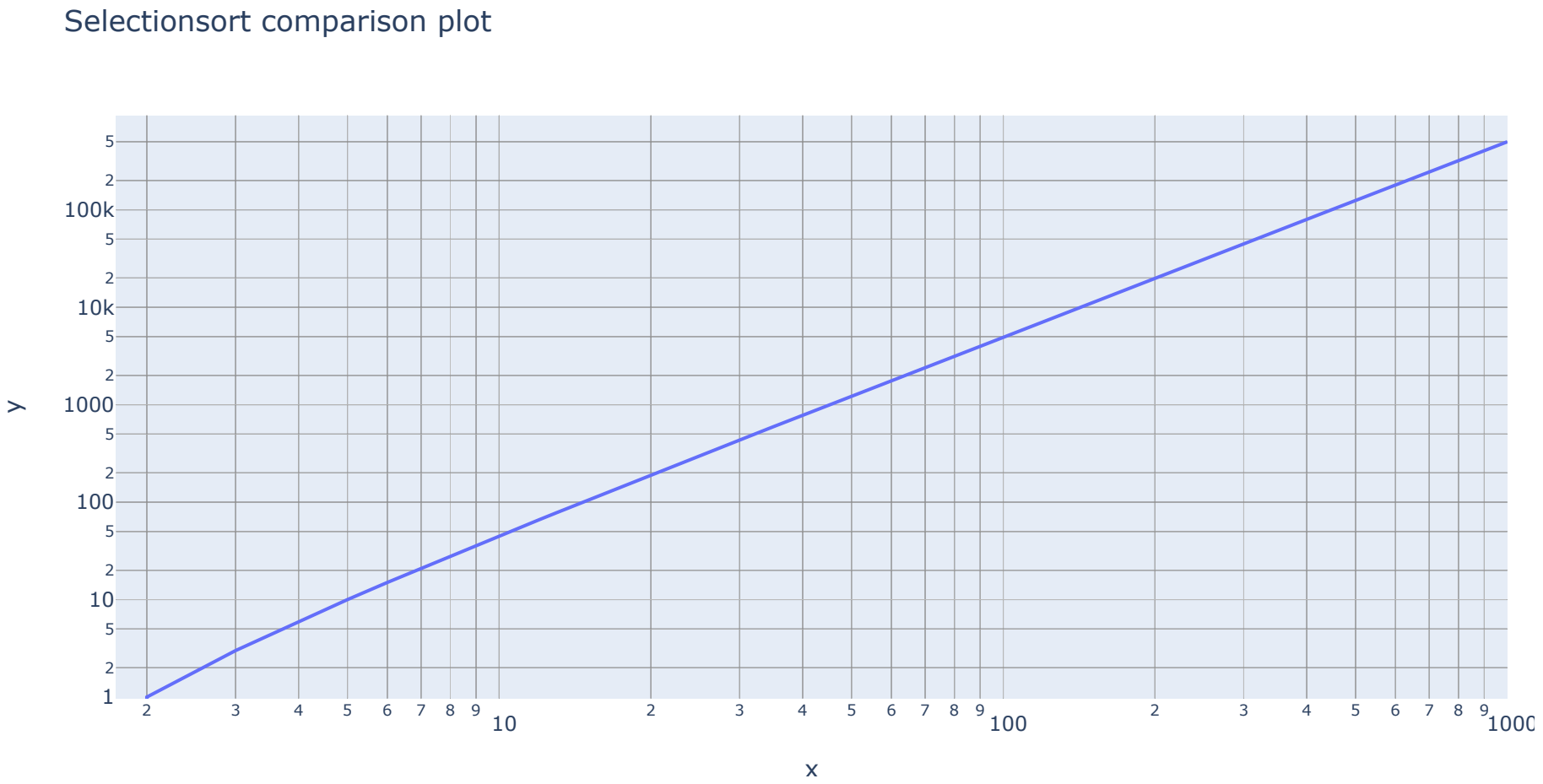

**Insertion sort plot:**

In [14]:
```
fig = px.line(df_is.loc[:, df_is.columns != "index"], x= df_is['size of array'], y = df_is['amount of compares'], title=
fig.show()
```



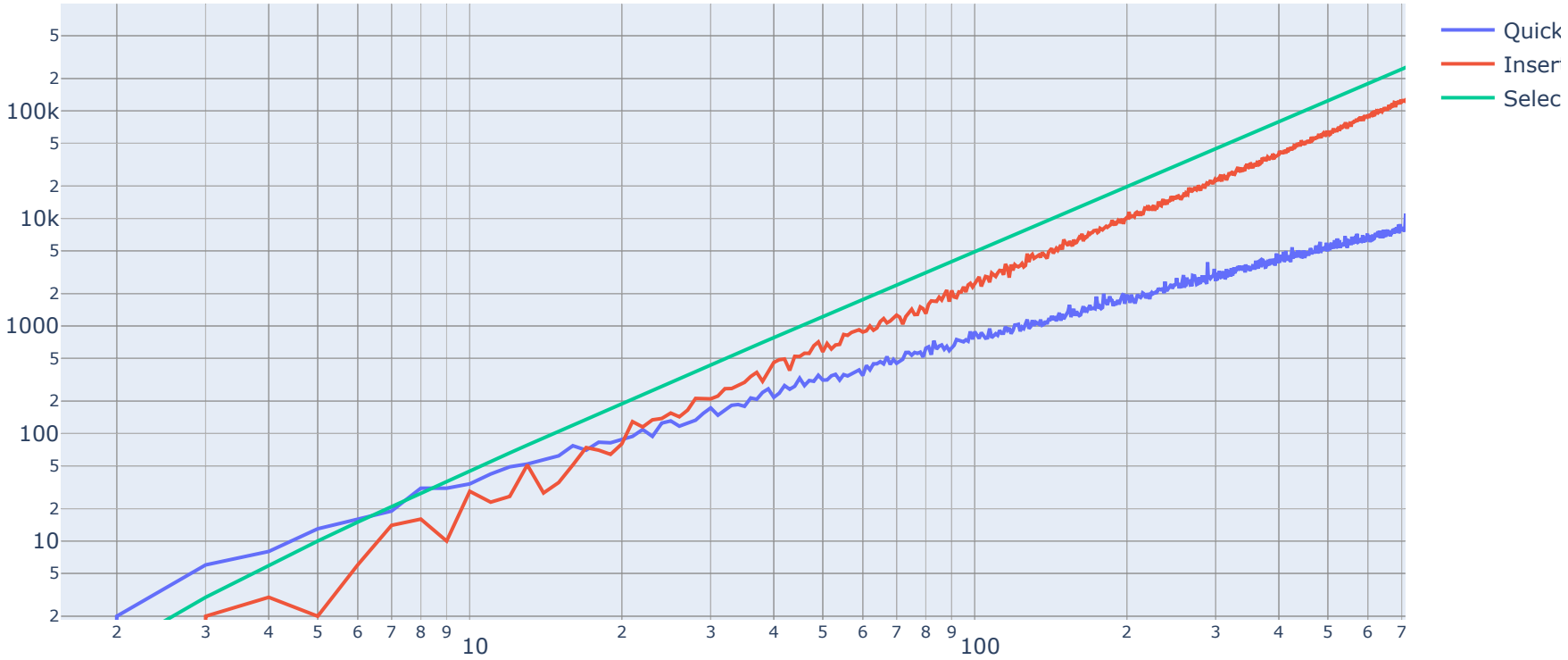

**Selection sort plot:**

In [9]:
```
fig = px.line(df_ss.loc[:, df_ss.columns != "index"], x= df_ss['size of array'], y = df_ss['amount of compares'], title=
fig.show()
```




**All algorithms together:**

In [16]:
```
fig = go.Figure()
fig.add_scatter(x =df_qs['size of array'], y = df_qs['amount of compares'], name = "Quicksort")
fig.add_scatter(x= df_is['size of array'], y = df_is['amount of compares'], name = "Insertionsort")
fig.add_scatter(x= df_ss['size of array'], y = df_ss['amount of compares'], name = "Selectionsort")
fig.update_xaxes(type="log")
fig.update_yaxes(type="log")
fig.show()
```



# Plotting and visualizing the Doubling-ratio experiment

## Reading the data

In [86]:
```
df_qs_ = pd.read_csv('./quicksort_dbr.csv')
df_is_ = pd.read_csv('./insertionsort_dbr.csv')
```

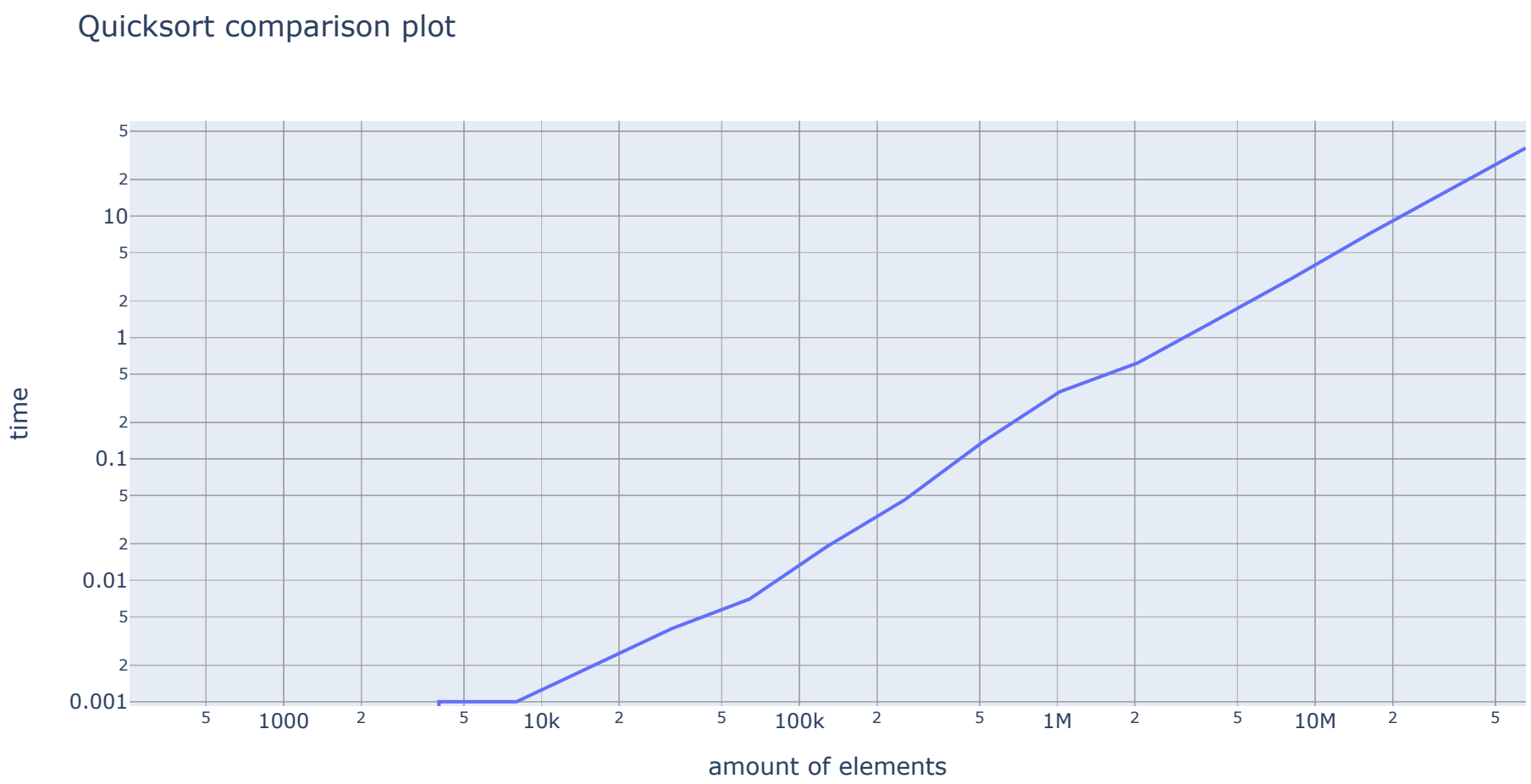
In [87]:
```
df_qs_.dbr = df_qs_.dbr.shift(-1).fillna('/')
```

In [88]:
```
df_qs_
```

Out[88]:

| | index | amount of elements | time | dbr |
|---|---|---|---|---|
| **0** | 0 | 250 | 0.000 | 0 |
| **1** | 1 | 500 | 0.000 | 0 |
| **2** | 2 | 1000 | 0.000 | 0 |
| **3** | 3 | 2000 | 0.000 | 0 |
| **4** | 4 | 4000 | 0.001 | 0 |
| **5** | 5 | 8000 | 0.001 | 1 |
| **6** | 6 | 16000 | 0.002 | 2 |
| **7** | 7 | 32000 | 0.004 | 2 |
| **8** | 8 | 64000 | 0.007 | 1.75 |
| **9** | 9 | 128000 | 0.019 | 2.71429 |
| **10** | 10 | 256000 | 0.046 | 2.42105 |
| **11** | 11 | 512000 | 0.137 | 2.97826 |
| **12** | 12 | 1024000 | 0.358 | 2.61314 |
| **13** | 13 | 2048000 | 0.617 | 1.72346 |
| **14** | 14 | 4096000 | 1.378 | 2.23339 |
| **15** | 15 | 8192000 | 3.101 | 2.25036 |
| **16** | 16 | 16384000 | 7.244 | 2.33602 |
| **17** | 17 | 32768000 | 16.051 | 2.21576 |
| **18** | 18 | 65536000 | 36.391 | / |

In [96]:
```
fig= px.line(df_qs_.loc[:, df_qs_.columns != "index"], x= df_qs_['amount of elements'], y = df_qs_['time'], title='Quicks
fig.update_xaxes(title_text='amount of elements')
fig.update_yaxes(title_text='time')
fig.show()
```



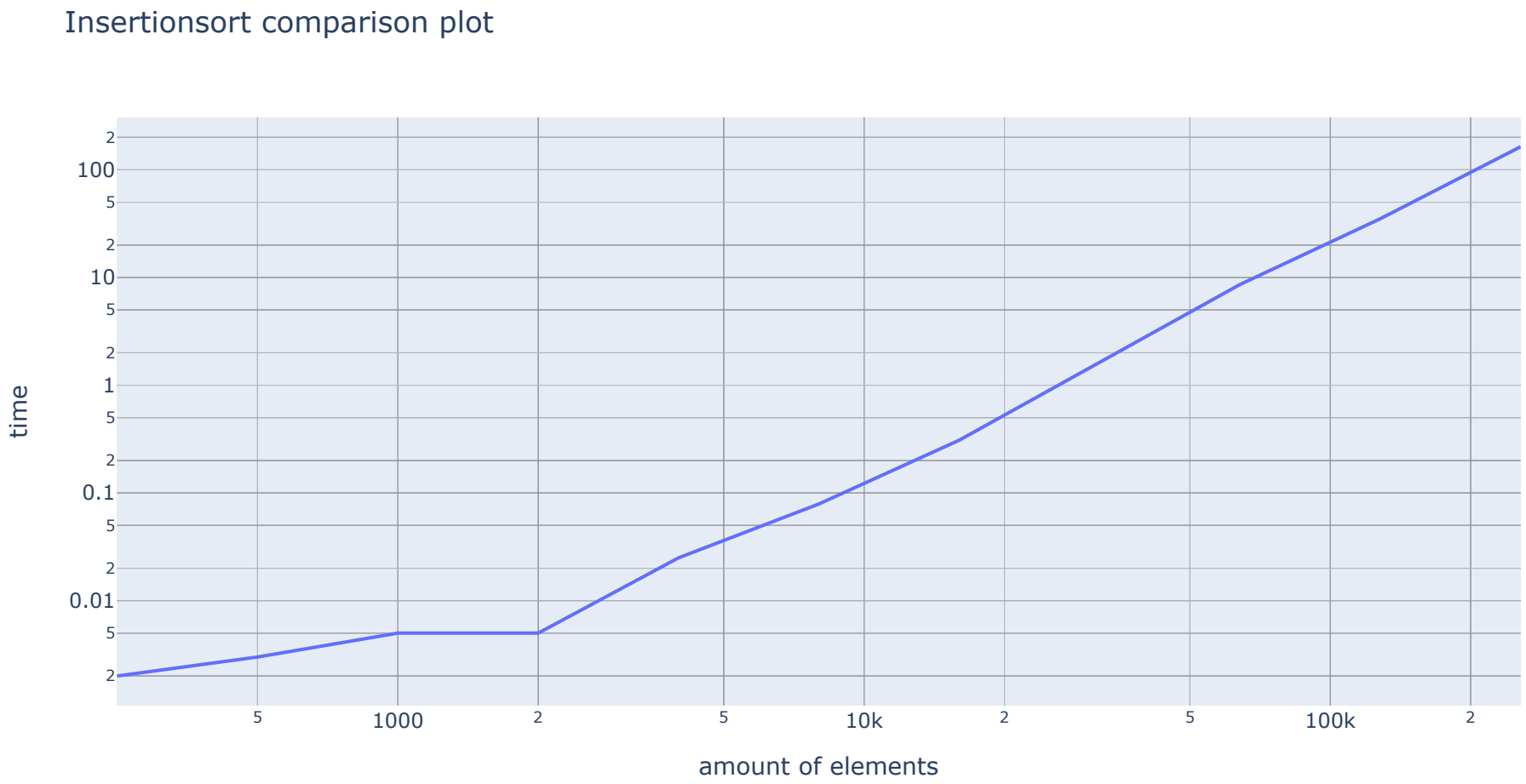

In [97]:
```
df_is_.dbr = df_is_.dbr.shift(-1).fillna('/')
```

In [98]:
```
df_is_
```

Out[98]:

| | index | amount of elements | time | dbr |
|---|---|---|---|---|
| **0** | 0 | 250 | 0.002 | 2 |
| **1** | 1 | 500 | 0.003 | 1.5 |
| **2** | 2 | 1000 | 0.005 | 1.66667 |
| **3** | 3 | 2000 | 0.005 | 1 |
| **4** | 4 | 4000 | 0.025 | 5 |
| **5** | 5 | 8000 | 0.079 | 3.16 |
| **6** | 6 | 16000 | 0.309 | 3.91139 |
| **7** | 7 | 32000 | 1.639 | 5.30421 |
| **8** | 8 | 64000 | 8.609 | 5.25259 |
| **9** | 9 | 128000 | 35.099 | 4.07701 |
| **10** | 10 | 256000 | 163.677 | / |

In [95]:
```
fig = px.line(df_is_.loc[:, df_is_.columns != "index"], x= df_is_['amount of elements'], y = df_is_['time'], title='Inse
fig.update_xaxes(title_text='amount of elements')
fig.update_yaxes(title_text='time')
fig.show()
```




In [ ]:
```

```