Mahesh Someshwar Patil

RBT21CB062

Topic: Analysis of Time complexities of Sorting Algorithms

```
import plotly #used for plotting graphs
import pandas as pd
import plotly.express as px
```

```
#creating dataframe Best Case
df1=pd.DataFrame([[10,1,"Insertion"],[100,1,"Insertion"],[500,2,"Insertion"],[1000,3,"Insertion"],
                  [2500,4,"Insertion"],[5000,6,"Insertion"],[7500,8,"Insertion"],[10000,13,"Insertion"],
                  [10,1,"Selection"],[100,6,"Selection"],[500,122,"Selection"],[1000,350,"Selection"],
                  [2500,2038,"Selection"],[5000,10288,"Selection"],[7500,19766,"Selection"],[10000,34465,"Selection"],
                  [10,1,"Quick"],[100,8,"Quick"],[500,121,"Quick"],[1000,451,"Quick"],
                  [2500,1941,"Quick"],[5000,9597,"Quick"],[7500,18402,"Quick"],[10000,30486,"Quick"]],
                 columns=["Range","Time","Sorting"])
df1
```

	Range	Time	Sorting
0	10	1	Insertion
1	100	1	Insertion
2	500	2	Insertion
3	1000	3	Insertion
4	2500	4	Insertion
5	5000	6	Insertion
6	7500	8	Insertion
7	10000	13	Insertion
8	10	1	Selection
9	100	6	Selection
10	500	122	Selection
11	1000	350	Selection
12	2500	2038	Selection
13	5000	10288	Selection
14	7500	19766	Selection
15	10000	34465	Selection
16	10	1	Quick
17	100	8	Quick
18	500	121	Quick
19	1000	451	Quick
20	2500	1941	Quick
21	5000	9597	Quick
22	7500	18402	Quick
23	10000	30486	Quick

```
df2=pd.DataFrame([[10,1,"Insertion"],[100,7,"Insertion"],[500,87,"Insertion"],[1000,357,"Insertion"],
                  [2500,1280,"Insertion"],[5000,4769,"Insertion"],[7500,15916,"Insertion"],[10000,34635,"Insertion"],
                  [10,1,"Selection"],[100,8,"Selection"],[500,126,"Selection"],[1000,450,"Selection"],
                  [2500,2105,"Selection"],[5000,10737,"Selection"],[7500,21637,"Selection"],[10000,44803,"Selection"],
                  [10,1,"Quick"],[100,5,"Quick"],[500,38,"Quick"],[1000,52,"Quick"],
                  [2500,131,"Quick"],[5000,332,"Quick"],[7500,569,"Quick"],[10000,765,"Quick"]],
                 columns=["Range","Time","Sorting"])
df2
```

	Range	Time	Sorting
0	10	1	Insertion
1	100	7	Insertion
2	500	87	Insertion
3	1000	357	Insertion
4	2500	1280	Insertion
5	5000	4769	Insertion
6	7500	15916	Insertion
7	10000	34635	Insertion
8	10	1	Selection
9	100	8	Selection
10	500	126	Selection
11	1000	450	Selection
12	2500	2105	Selection
13	5000	10737	Selection
14	7500	21637	Selection
15	10000	44803	Selection
16	10	1	Quick
17	100	5	Quick
18	500	38	Quick
19	1000	52	Quick
20	2500	131	Quick
21	5000	332	Quick
22	7500	569	Quick

```
df3=pd.DataFrame([[10,1,"Insertion"],[100,8,"Insertion"],[500,113,"Insertion"],[1000,437,"Insertion"],
                  [2500,2284,"Insertion"],[5000,8430,"Insertion"],[7500,19182,"Insertion"],[10000,48668,"Insertion"],
                  [10,1,"Selection"],[100,10,"Selection"],[500,131,"Selection"],[1000,466,"Selection"],
                  [2500,2290,"Selection"],[5000,9677,"Selection"],[7500,20925,"Selection"],[10000,50787,"Selection"],
                  [10,1,"Quick"],[100,7,"Quick"],[500,92,"Quick"],[1000,396,"Quick"],
                  [2500,1804,"Quick"],[5000,5799,"Quick"],[7500,13768,"Quick"],[10000,24788,"Quick"]],
                 columns=["Range","Time","Sorting"])
df3
```

	Range	Time	Sorting
0	10	1	Insertion
1	100	8	Insertion
2	500	113	Insertion
3	1000	437	Insertion
4	2500	2284	Insertion
5	5000	8430	Insertion
6	7500	19182	Insertion

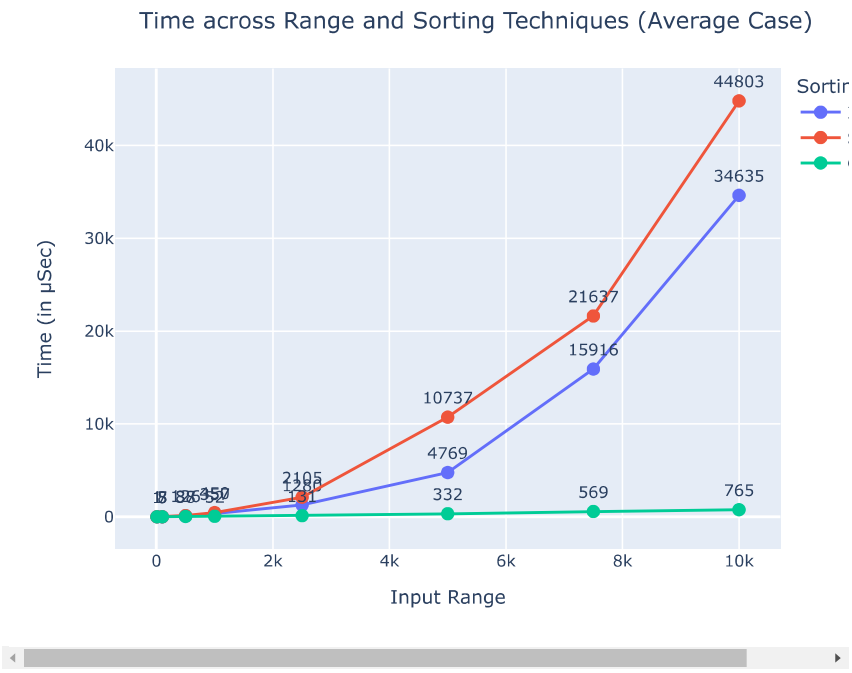
Sorting Algorithms	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	Worst Case
Bubble Sort	Ω(N)	Θ(N^2)	O(N^2)	O(1)
Selection Sort	Ω(N^2)	Θ(N^2)	O(N^2)	O(1)
Insertion Sort	Ω(N)	Θ(N^2)	O(N^2)	O(1)
Quick Sort	Ω(N log N)	Θ(N log N)	O(N^2)	O(N)

20	2500	1804	Quick

```
fig1=px.line(df1,x="Range",y="Time",#template="plotly_dark",
             width=700,height=500,
            labels={"Time":"Time (in µSec)","Range":"Input Range"},markers=True,text="Time",color="Sorting")
fig1.update_traces(textposition="top center")
fig1.update_traces(marker_size=10)
fig1.update_layout(title_text="Time across Range and Sorting Techniques (Best Case)",title_x=0.5)
fig1.show()
```


```
fig2=px.line(df2,x="Range",y="Time",width=700,height=500,
            labels={"Time":"Time (in µSec)","Range":"Input Range"},markers=True,text="Time",color="Sorting")
fig2.update_traces(textposition="top center")
fig2.update_traces(marker_size=10)
```

```
fig2.update_layout(title_text="Time across Range and Sorting Techniques (Average Case)",title_x=0.5)
fig2.show()
```




```
fig3=px.line(df3,x="Range",y="Time",#template="plotly_dark",
             width=700,height=500,
            labels={"Time":"Time (in µSec)","Range":"Input Range"},markers=True,text="Time",color="Sorting")
fig3.update_traces(textposition="top center")
fig3.update_traces(marker_size=10)
fig3.update_layout(title_text="Time across Range and Sorting Techniques (Worst Case)",title_x=0.5)
fig3.show()
```

Conclusion : Results are interpreted correctly by analyzing the observations.