Ahmet Faruk Ulutaş - 21803717 - CS 342 Project 1

I have automatically generated test files using the code below.

```
import random
from faker import Faker

# Define the values for N and K that you want to test
Ns = [2, 4, 6, 8, 10]
Ks = [10, 100, 1000, 10000]

# Define the number of files you want to create
num_files = 10

# Define the length of each file
file_length = 10000

# Initialize the Faker library
fake = Faker()

# Generate the files
for i in range(num_files):
        # Create a new file
        filename = f"file_{i}.txt"
        with open(filename, "w") as f:
        # Add random words to the text
        for _ in range(file_length):
        f.write(fake.word() + " ")

        # Print a message indicating that the file has been created
        print(f"File '{filename}' created.")
```

PROCTOPK OUTPUT

```
Command './proctopk 10 outfile.txt 2 file_0.txt file_1.txt' took 0.0074 seconds to execute.
Command './proctopk 10 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.0112 seconds to execute.
Command './proctopk 10 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt' took 0.0157 seconds to execute.
Command './proctopk 10 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt' took 0.0339 seconds to execute.
Command './proctopk 10 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.0224 seconds to execute.

Command './proctopk 100 outfile.txt 2 file_0.txt file_1.txt' took 0.0327 seconds to execute.
Command './proctopk 100 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.0464 seconds to execute.
Command './proctopk 100 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt' took 0.0531 seconds to execute.
Command './proctopk 100 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt
file_6.txt file_7.txt' took 0.0526 seconds to execute.
Command './proctopk 100 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.0789 seconds to execute.

Command './proctopk 1000 outfile.txt 2 file_0.txt file_1.txt' took 0.0675 seconds to execute.
Command './proctopk 1000 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.1019
seconds to execute.
Command './proctopk 1000 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt' took 0.2135 seconds to execute.
Command './proctopk 1000 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt' took 0.3339 seconds to execute.
Command './proctopk 1000 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.4897 seconds to execute.
Segmentation fault (core dumped)

Command './proctopk 10000 outfile.txt 2 file_0.txt file_1.txt' took 0.2895 seconds to execute.
Segmentation fault (core dumped)
Command './proctopk 10000 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.3167
seconds to execute.
Segmentation fault (core dumped)
Command './proctopk 10000 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt' took 0.3293 seconds to execute.
Segmentation fault (core dumped)
Command './proctopk 10000 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt' took 0.3853 seconds to execute.
Segmentation fault (core dumped)
Command './proctopk 10000 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.5082 seconds to execute.
```

THREADTOPK OUTPUT

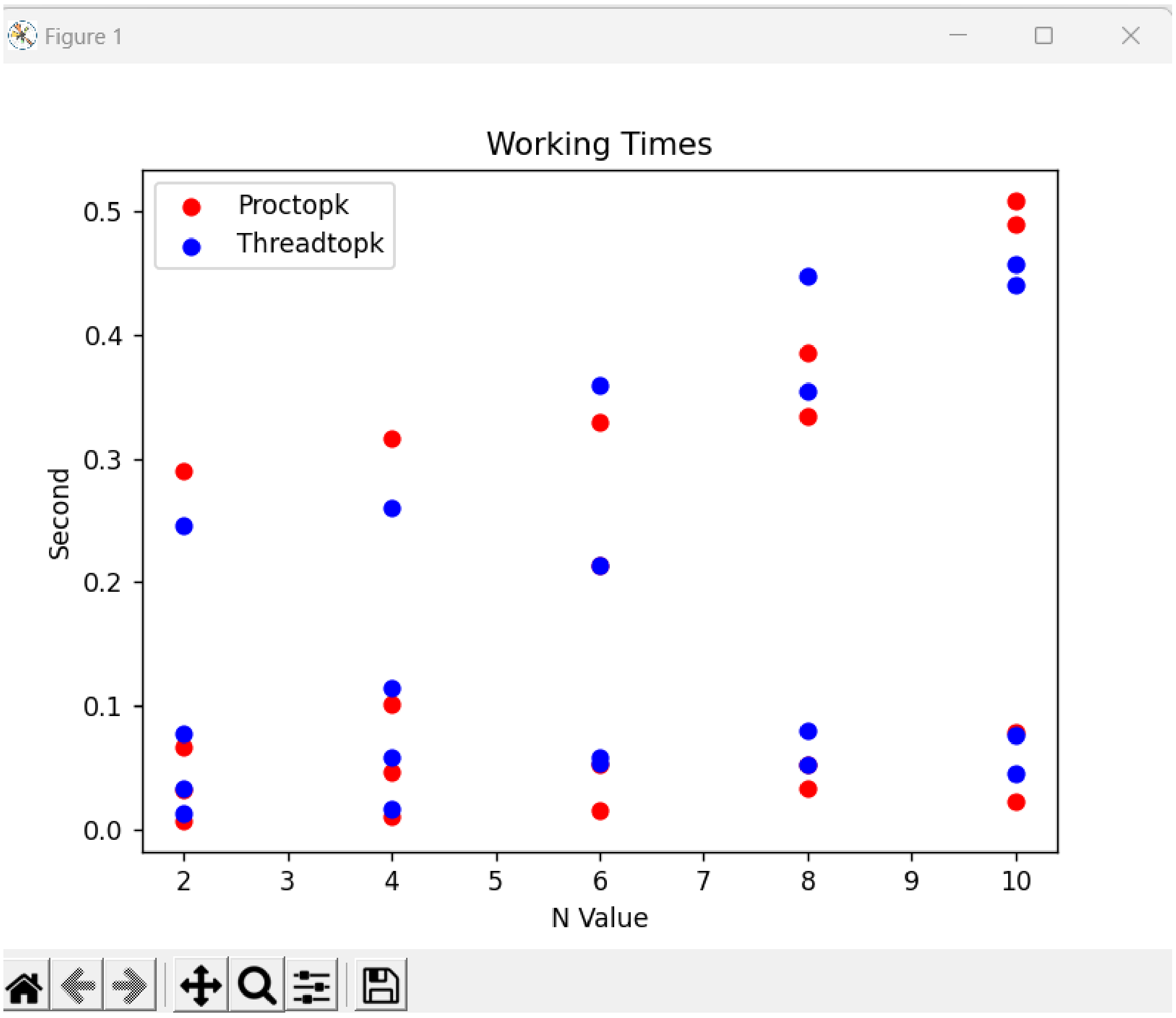
```
Command './threadtopk 10 outfile.txt 2 file_0.txt file_1.txt' took 0.0133 seconds to execute.
Command './threadtopk 10 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.0166
seconds to execute.
Command './threadtopk 10 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt' took 0.0587 seconds to execute.
Command './threadtopk 10 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt
file_6.txt file_7.txt' took 0.0528 seconds to execute.
Command './threadtopk 10 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.0450 seconds to execute.

Command './threadtopk 100 outfile.txt 2 file_0.txt file_1.txt' took 0.0332 seconds to execute.
Command './threadtopk 100 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.0581
seconds to execute.
Command './threadtopk 100 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt' took 0.0543 seconds to execute.
Command './threadtopk 100 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt
file_5.txt file_6.txt file_7.txt' took 0.0799 seconds to execute.
```

```
Command './threadtopk 100 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.0765 seconds to execute.

Command './threadtopk 1000 outfile.txt 2 file_0.txt file_1.txt' took 0.0778 seconds to execute.
Command './threadtopk 1000 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.1144 seconds to execute.
Command './threadtopk 1000 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt' took 0.2134 seconds to execute.
Command './threadtopk 1000 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt' took 0.3547 seconds to execute.
Command './threadtopk 1000 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.4572 seconds to execute.
Segmentation fault (core dumped)

Command './threadtopk 10000 outfile.txt 2 file_0.txt file_1.txt' took 0.2461 seconds to execute.
Segmentation fault (core dumped)
Command './threadtopk 10000 outfile.txt 4 file_0.txt file_1.txt file_2.txt file_3.txt' took 0.2599 seconds to execute.
Segmentation fault (core dumped)
Command './threadtopk 10000 outfile.txt 6 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt' took 0.3590 seconds to execute.
Segmentation fault (core dumped)
Command './threadtopk 10000 outfile.txt 8 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt' took 0.4470 seconds to execute.
Segmentation fault (core dumped)
Command './threadtopk 10000 outfile.txt 10 file_0.txt file_1.txt file_2.txt file_3.txt file_4.txt file_5.txt file_6.txt file_7.txt file_8.txt file_9.txt' took 0.4406 seconds to execute.
```

I created the chart automatically using the code below.

```
import matplotlib.pyplot as plt

proctopk_n_values = [2, 4, 6, 8, 10, 2, 4, 6, 8, 10, 2, 4, 6, 8, 10, 2, 4, 6, 8, 10]
proctopk_saniyeler = [0.0074, 0.0112, 0.0157, 0.0339, 0.0224, 0.0327, 0.0464, 0.0531, 0.0526, 0.0789, 0.0675, 0.1019, 0.2135, 0.3339, 0.4897, 0.2895, 0.3167, 0.3293, 0.3853, 0.5082]

threadtopk_n_values = [2, 4, 6, 8, 10, 2, 4, 6, 8, 10, 2, 4, 6, 8, 10, 2, 4, 6, 8, 10]
threadtopk_saniyeler = [0.0133, 0.0166, 0.0587, 0.0528, 0.0450, 0.0332, 0.0581, 0.0543, 0.0799, 0.0765, 0.0778, 0.1144, 0.2134, 0.3547, 0.4572, 0.2461, 0.2599, 0.3590, 0.4470, 0.4406]

# Boyutları eşitlemek için gereksiz verileri sil
del proctopk_saniyeler[len(proctopk_n_values):]
del threadtopk_saniyeler[len(threadtopk_n_values):]

proctopk_color = 'red'
```

```
threadtopk_color = 'blue'

plt.scatter(proctopk_n_values, proctopk_saniyeler, color=proctopk_color, label='Proctopk')
plt.scatter(threadtopk_n_values,          threadtopk_saniyeler,          color=threadtopk_color,
label='Threadtopk')

plt.title('Working Times')
plt.xlabel('N Value')
plt.ylabel('Second')
plt.legend()
plt.show()
```



RESULT
The analysis's findings show that as file size and word count grow, so does the running duration of both applications. The threadtopk program, however, operates more quickly than the proctopk program. In particular, threadtopk clearly outperforms proctopk as the quantity of files and lines increases.

Program failures were another problem that came up and significantly extended the duration of programs. This problem may be caused by memory leakage, a lack of process synchronisation, or other issues of a similar nature. Programs must therefore be correctly tuned.