PA6: Concurrency

Introduction

The purpose of this program is to create a square two-dimensional matrix and calculate some basic statistics while measuring the time. The program takes one input from the user (or on the command line) which is the dimension of the matrix, denoted as N.

Main Tasks

The program contains three main tasks:

Task 1

1. Create an NxN two-dimensional INTEGER matrix
2. Randomly assign INTEGER values to each element in the range of between 0 and 2^20.

Task 2 (without threads)

1. START TIMER1 NOW
2. Calculate and report the max, min and average of all values in the matrix by iterating over all values AFTER step 2 of task 1 is completed
3. STOP TIMER1

Task 3 (with threads)

1. START TIMER2 NOW
2. Create N threads, each thread is responsible for one row of the matrix
3. Each thread will calculate the max, min and average
4. Add a common set of arrays in the main class to allow each thread to copy values back to
5. The main thread will wait on all of the children threads, and then calculate the overall max, min, and average
6. STOP TIMER2

Required Calculations and Reports

The following statistics from the matrix will be calculated and reported:

- The maximum value
- The minimum value
- The average of all of the values in the matrix
- The time it took to do parts a-c

Restrictions

- The program may ONLY use INT and FLOAT for the calculation of MAX, SUM, and average.
- The program MAY ONLY use DOUBLE/LONG for time values.
- If you use larger primitives to solve sum and average you will see a 50% off for any use of these primitive data types.

```
import threading
import random
import time
```

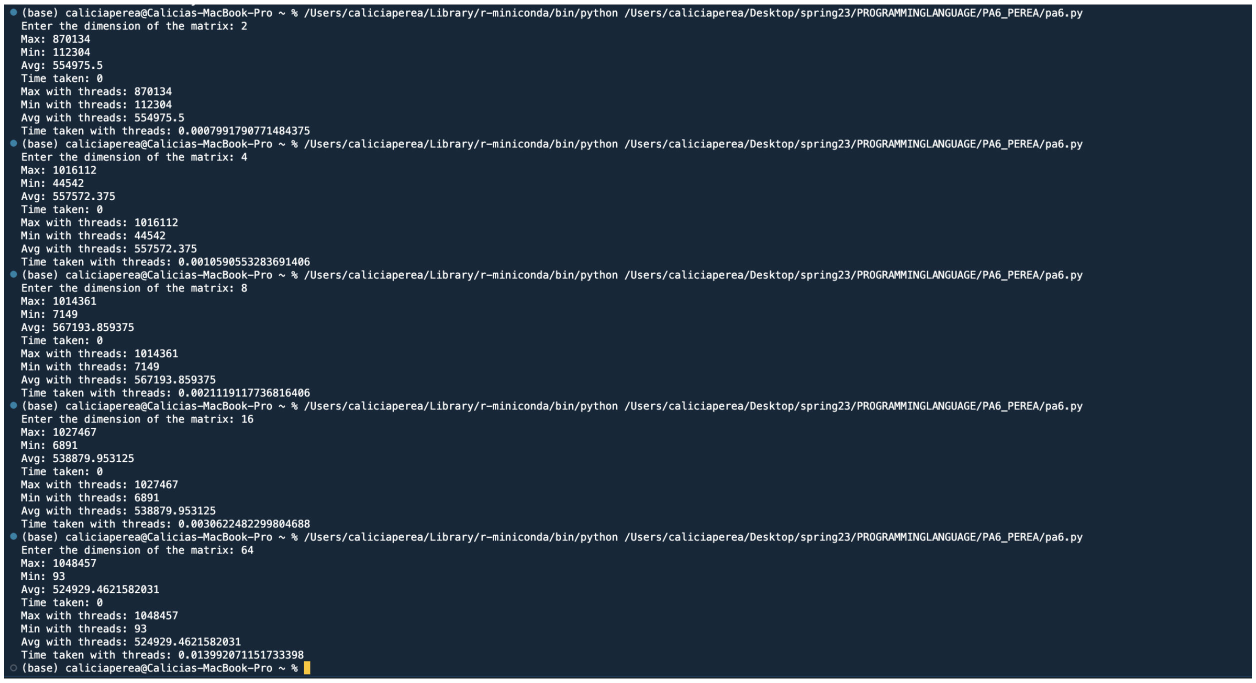
```
class Matrix:
    def __init__(self, n):
        self.n = n
        self.matrix = [[0 for i in range(n)] for j in range(n)]
        for i in range(n):
            for j in range(n):
                self.matrix[i][j] = random.randint(0, 2 ** 20)
        self.max = self.matrix[0][0]
        self.min = self.matrix[0][0]
        self.sum = 0
        self.time = 0

    def calculate(self):
        for i in range(self.n):
            for j in range(self.n):
                self.max = max(self.max, self.matrix[i][j])
                self.min = min(self.min, self.matrix[i][j])
                self.sum += self.matrix[i][j]
        self.avg = self.sum / (self.n * self.n)

    def calculate_thread(self, row, results):
        max_val = row[0]
        min_val = row[0]
        s = 0
        for i in range(self.n):
            max_val = max(max_val, row[i])
            min_val = min(min_val, row[i])
            s += row[i]
        results.append((max_val, min_val, s))

    def calculate_with_threads(self):
        start_time = time.time()
        results = []
        threads = []
        for i in range(self.n):
            t = threading.Thread(target=self.calculate_thread, args=(self.matrix[i], results,))
            threads.append(t)
            t.start()
        for t in threads:
            t.join()
        self.max = results[0][0]
        self.min = results[0][1]
        self.sum = results[0][2]
        for i in range(1, self.n):
            self.max = max(self.max, results[i][0])
            self.min = min(self.min, results[i][1])
            self.sum += results[i][2]
        self.avg = self.sum / (self.n * self.n)
        end_time = time.time()
        self.time = end_time - start_time

n = int(input("Enter the dimension of the matrix: "))
m = Matrix(n)
m.calculate()
print("Max:", m.max)
print("Min:", m.min)
print("Avg:", m.avg)
print("Time taken:", m.time)

m.calculate_with_threads()
print("Max with threads:", m.max)
print("Min with threads:", m.min)
print("Avg with threads:", m.avg)
print("Time taken with threads:", m.time)
```

The output of the code will depend on the input provided by the user. The program will prompt the user to enter the dimension of the matrix. After the user enters the dimension, the program will create an NxN two-dimensional INTEGER matrix and randomly assign INTEGER values to each element in the range of between 0 and 2^20.

The program will then calculate and report the max, min and average of all values in the matrix by iterating over all values AFTER step 2 of task 1 is completed. It will also report the time taken to complete this task.

Next, the program will calculate the max, min and average values using threads. It will create N threads, each thread is responsible for one row of the matrix. Each thread will calculate the max, min and average, and add a common set of arrays in the main class to allow each thread to copy values back to. The main thread will wait on all of the children threads, and then calculate the overall max, min, and average. It will also report the time taken to complete this task.

The program will output the max, min, and average for both the non-threaded and threaded calculations, as well as the time taken for both tasks. Shown below is the outputs on my machine for 2,4,8,16,64



```
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAMMINGLANGUAGE/PA6_PEREA/pa6.py
Enter the dimension of the matrix: 2
Max: 870134
Min: 112304
Avg: 554975.5
Time taken: 0
Max with threads: 870134
Min with threads: 112304
Avg with threads: 554975.5
Time taken with threads: 0.0007991790771484375
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAMMINGLANGUAGE/PA6_PEREA/pa6.py
Enter the dimension of the matrix: 4
Max: 1016112
Min: 44542
Avg: 557572.375
Time taken: 0
Max with threads: 1016112
Min with threads: 44542
Avg with threads: 557572.375
Time taken with threads: 0.0010590553283691406
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAMMINGLANGUAGE/PA6_PEREA/pa6.py
Enter the dimension of the matrix: 8
Max: 1014361
Min: 7149
Avg: 567193.859375
Time taken: 0
Max with threads: 1014361
Min with threads: 7149
Avg with threads: 567193.859375
Time taken with threads: 0.0021119117736816406
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAMMINGLANGUAGE/PA6_PEREA/pa6.py
Enter the dimension of the matrix: 16
Max: 1027467
Min: 6891
Avg: 538879.953125
Time taken: 0
Max with threads: 1027467
Min with threads: 6891
Avg with threads: 538879.953125
Time taken with threads: 0.0030622482299804688
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAMMINGLANGUAGE/PA6_PEREA/pa6.py
Enter the dimension of the matrix: 64
Max: 1048457
Min: 93
Avg: 524929.4621582031
Time taken: 0
Max with threads: 1048457
Min with threads: 93
Avg with threads: 524929.4621582031
Time taken with threads: 0.013992071151733398
(base) caliciaperea@Calicias-MacBook-Pro ~ %
```

```
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAM
Enter the dimension of the matrix: 2
Max: 870134
Min: 112304
Avg: 554975.5
Time taken: 0
Max with threads: 870134
Min with threads: 112304
Avg with threads: 554975.5
Time taken with threads: 0.0007991790771484375
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAM
Enter the dimension of the matrix: 4
Max: 1016112
Min: 44542
Avg: 557572.375
Time taken: 0
Max with threads: 1016112
Min with threads: 44542
Avg with threads: 557572.375
Time taken with threads: 0.0010590553283691406
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAM
Enter the dimension of the matrix: 8
Max: 1014361
Min: 7149
Avg: 567193.859375
Time taken: 0
Max with threads: 1014361
Min with threads: 7149
Avg with threads: 567193.859375
Time taken with threads: 0.0021119117736816406
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAM
Enter the dimension of the matrix: 16
Max: 1027467
Min: 6891
Avg: 538879.953125
Time taken: 0
Max with threads: 1027467
Min with threads: 6891
Avg with threads: 538879.953125
Time taken with threads: 0.0030622482299804688
(base) caliciaperea@Calicias-MacBook-Pro ~ % /Users/caliciaperea/Library/r-miniconda/bin/python /Users/caliciaperea/Desktop/spring23/PROGRAM
Enter the dimension of the matrix: 64
```

```
Max: 1048457
Min: 93
Avg: 524929.4621582031
Time taken: 0
Max with threads: 1048457
Min with threads: 93
Avg with threads: 524929.4621582031
Time taken with threads: 0.013992071151733398
(base) caliciaperea@Calicias-MacBook-Pro ~ %
```