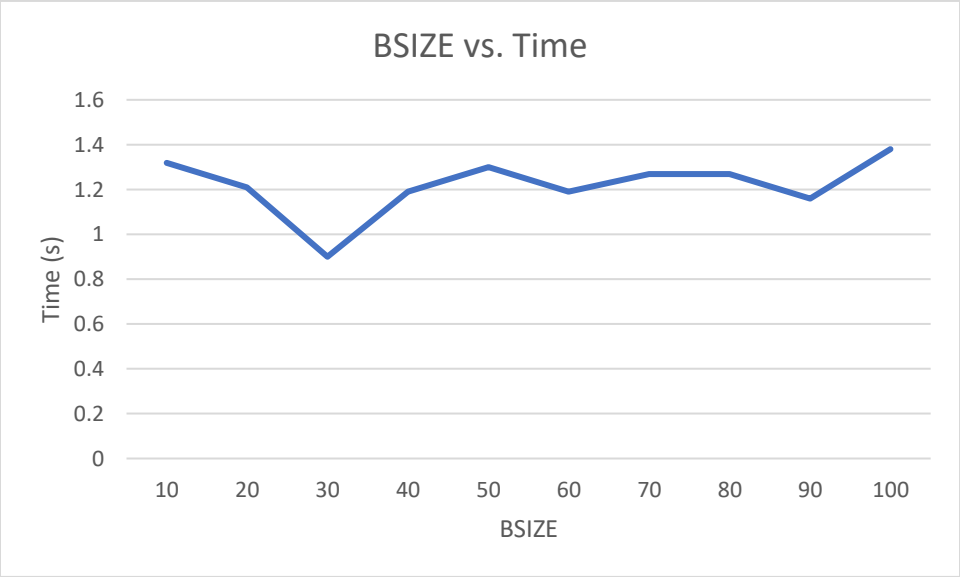
Suraj Ilavala

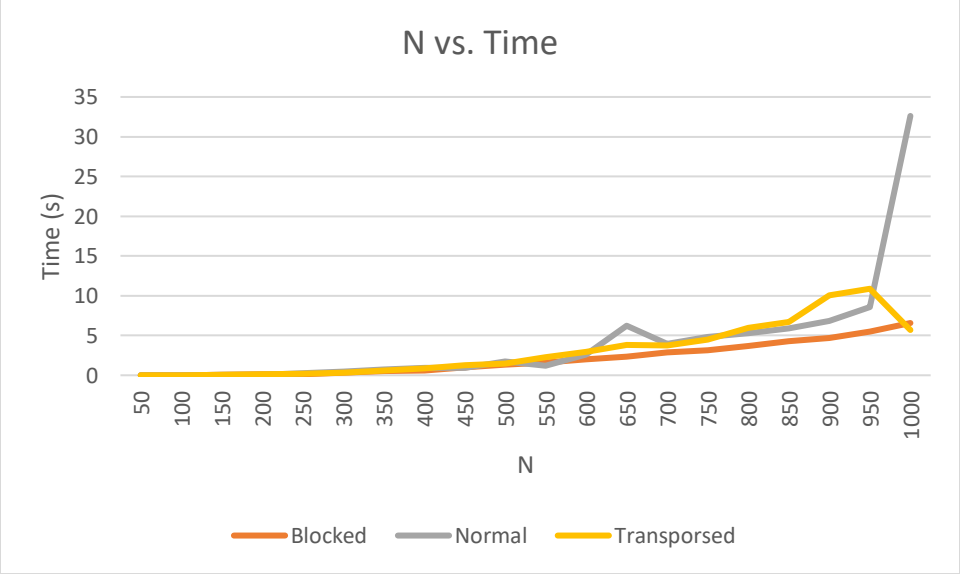
Assignment 4

BSIZE vs. Time



Blocked Equation = y = 0.0132x + 1.1467

N vs. Time



Normal Graph equation => y = 0.8201x - 4.4552

Blocked Graph equation => y = 0.3271x - 1.3938

Transposed Graph Equation => y = 0.5052x - 2.2272

Question 1)

The time returned correctly for algorithms in the blocked, normal, and transposed. When trying to run the Multithread block b memory could not be formed. Because it is trying to do to many processes at the same time.

Question 2)

With the test cases I chose the BSIZE 20 because it seemed a good pick. All the test cases had a consistent N number of 500. This is too keeping the check for BSIZE efficient. One thing I noticed is as BSIZE increases the shorter the time, but to see what is most efficient a program can be its best to keep the BSIZE the shortest. So, I have chosen 20 to be most ideal, this will keep it short and be able to see time variations between the N number.

Question 3)

Following the trend line the algorithms that performed the tasks the fastest is the blocked algorithm with 0.3271x. While others were almost double the efficiency, blocked was the most efficient. Using the ideal BSIZE, which is set at 20, it shows the block algorithm time is faster than any other algorithms.

Question 4)

For Normal matrix multiplication, the algorithm uses double to triple for loops to mallocate data. Matrices are indexed by [row][column] After the for loops take in its arguments it will then store the a total and multiplication of all argument in a appropriate location that was previously zeroed out. Its efficient because its big o notation is O(n^3). This is great for small numbers but as numbers increase it'll slow down the efficiency. Transposed is the same as the normal matrix but it indexed after Matrix is transposed. This is efficient because we do not need to worry about string until everything is done and made.

For Blocked, it uses 6 nested loops. 3 outer loops that increment though blocks and the inner 3 loops to perform multiplication of 1 block vs. another, and then adding it to a proper location. As the blocks are incremented, they are also stored at the same time. While normal is good, it separates what its supposed to do, but blocked does everything at once as it goes through.

Question 5)

When first looking at the description I thought the blocked algorithm was the fastest because as it increments it goes through the block. Its almost as if it is doing multiple processes as it goes through the for loops. The function is using a path and when it's at that path it does its business and continues. While the others separate these functions in different for loops.

Question 6)

I chose 20 because I want to keep the BSIZE small because it seemed the right size to be able to test the loops with bigger number. I chose this so that I can see the variations of time processed as n increases. If the BSIZE was too big it would be difficult to see the minor changes in time. So, choosing a small but not to small BSIZE will allow me to test N values and what algorithm is the most efficient.

Question 7)

To handle collisions, it makes the memory space before even adding to the memory. But for some reason when I tried running it didn't allow me to add to the multithread. It kept saying could not map b_memory. There seems to be some fault when trying to make up the secondary memory location for the next memory allocation set up. There seems to be some fault in the memory setup for the memory allocation.