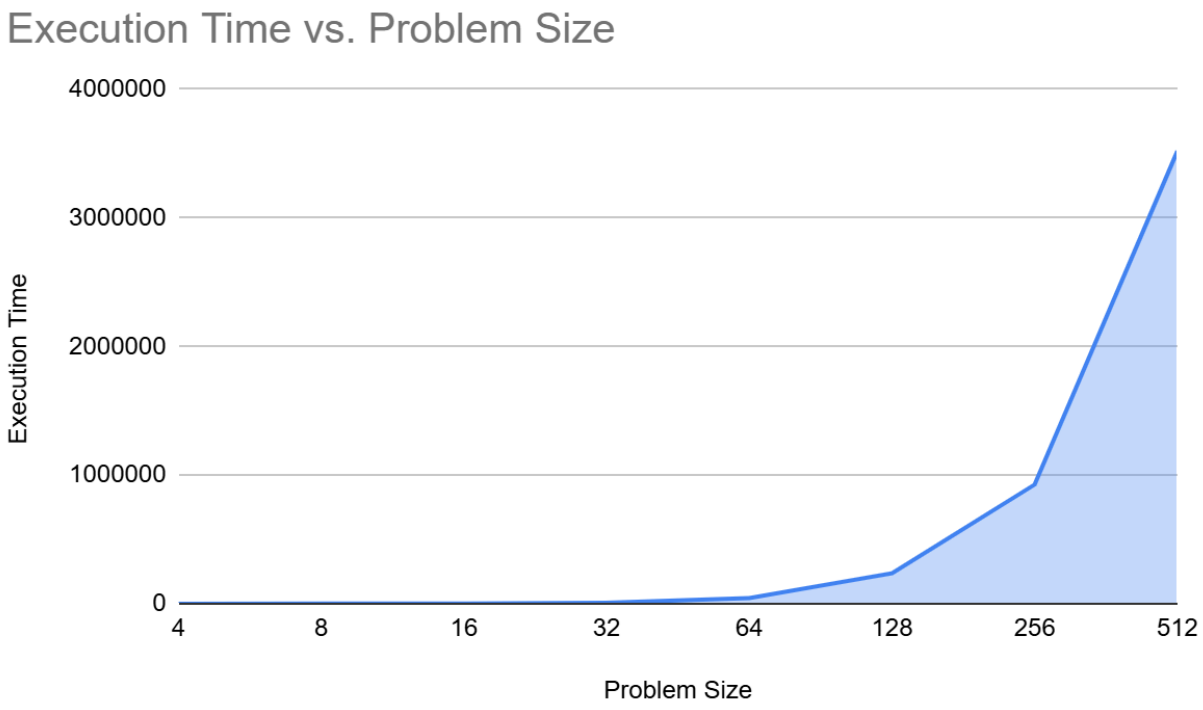
1.

(a) The kernel size is a 2x2 matrix. The reason I chose this is that this is the smallest recursive pattern for the z-ordering structure.

(b)



2.

(a) Assume the first column of A, the first row of B, and the first row of C all start mapping to set 0. These three data blocks will therefore occupy three different ways within that set. Matrix C contains 48 doubles, or 384 bytes in total, which is much smaller than the capacity of one way (4 KB). However, in the worst case, any additional data mapped to the same way as C could evict parts of it. To avoid such conflicts, we can conceptually reserve one entire way for C, leaving seven ways available for A and B. For the same reason, each of those seven ways should be dedicated to either A or B, but not shared between them. We can safely use three ways for A and another three for B, leaving one unused. Since each column of A contains 6 doubles (48 bytes) and each row of B contains 8 doubles (64 bytes), after filling three ways for B, A will have only used 2.75 ways. At that point, we can begin using the fourth way for A while B completes its third way. As a result, when k = 256, we have 4 ways allocated for A, 3 ways for B, and 1 way reserved for C, fully utilizing all eight ways without overwriting any data in C.

(b) In the previous arrangement, we used up 7 ways, and 384 bytes of the eighth way, which is a total of 29056 bytes. This is around 88.8% of the L1 cache.

(c) Since the L2 cache is 256 KB and 8-way set associative, each way can store up to 32 KB. This is enough to store everything in B. To avoid conflicts and evictions, we reserve one entire way for B and another for C, leaving six ways available for A. Each A matrix of size 6×256 contains 12 KB of data. With six remaining ways in L2, we have a total of 192 KB available for storing A, which is enough to store 16 such matrix. Because each matrix contributes 6 rows, we can stack 16 of them vertically. Therefore, the largest value of Mc is 96.

(d) We used up 6 ways, 16 KB of one way, and 384 B of the last way, which is 213376 bytes. This is about 81.4% of the L2 cache

(e) The program was able to achieve 13.360606 FLOPS/cycle, and the theoretical peak is 16 FLOPS/cycle. We are at around 83.5% of the theoretical peak.

3.

Around 4 hours.