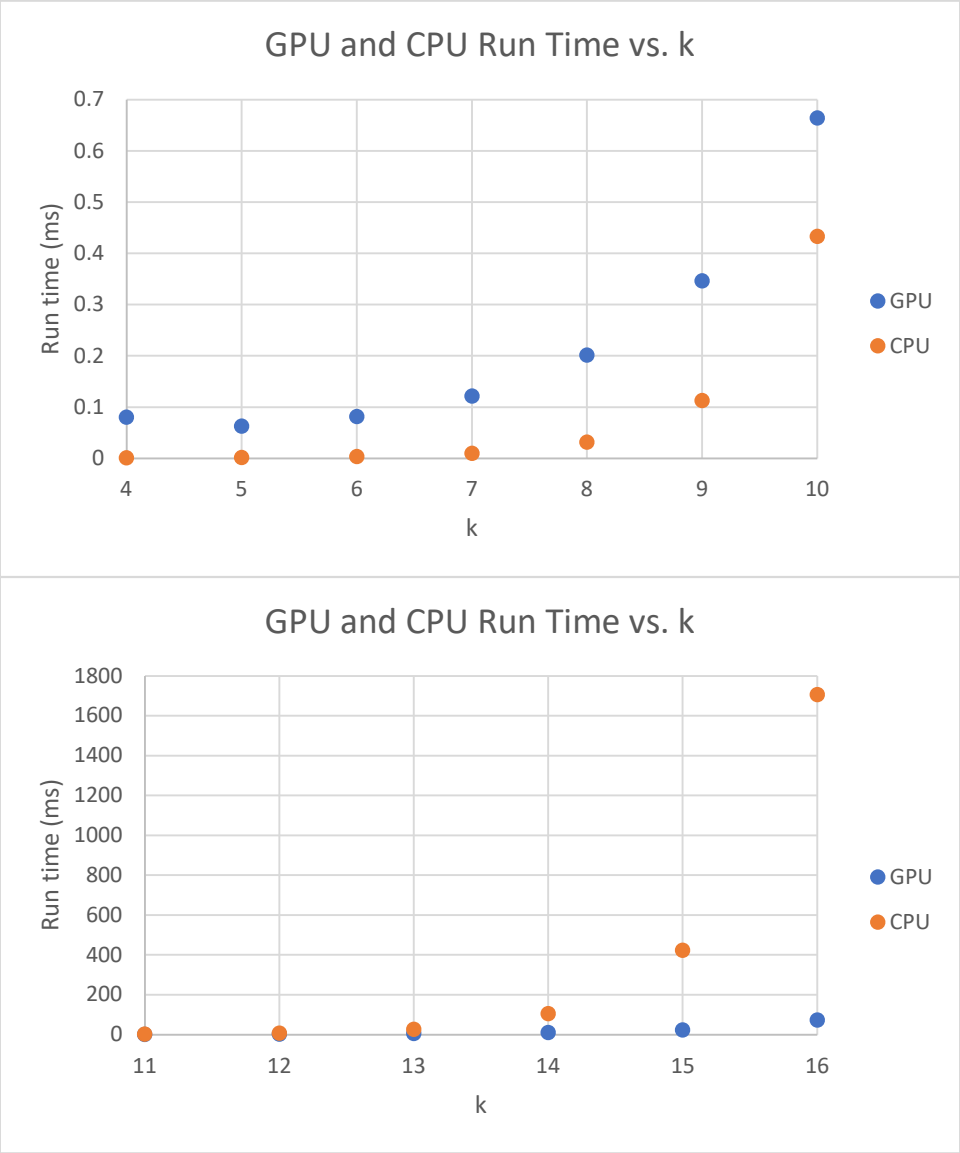
Jacob Hessong

HW5 Report

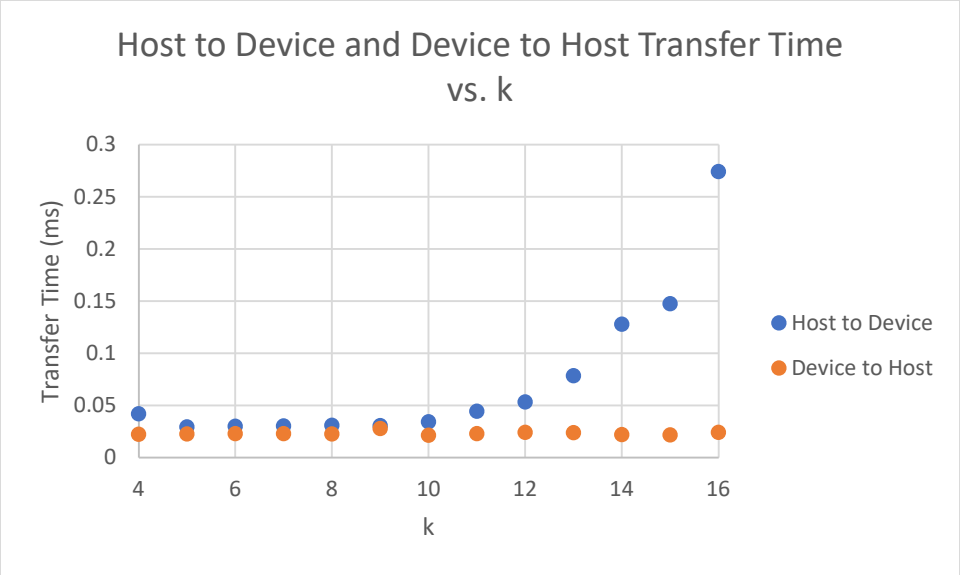
To perform the computation, I implemented the minimum_distance function as well as making a few changes to the main function. This function is called with n threads, with one thread per point. Threads were clustered into blocks of 1024 threads each. The modifications in main that help do this are: the creation of a new variable, num_blocks, that is the number of points divided by the block size plus the output of if the number of points mod block size is not equal to 0, and calling the minimum distance function using this number of blocks.

The minimum distance functions computes the thread's global id with respect to all threads using the equation blockIdx.x * blockDim.x + threadIdx.x. This helps with defining which point the thread will operate on. If a thread has an id larger than the number of points minus two, it will be an extra thread and not do anything. This helps with the case of the number of points minus two being less than the block size. Inside the if statement, each thread computes the distance between its point and all points with an index after it (does not compute distance with points before because it is redundant, as distance will be the same regardless of order i.e. distance from point 1 to 5 is the same as from 5 to 1). Each thread will then find the minimum distance of all threads within one block, as they cannot be synchronized with threads in a different block. At this step, there are block_size concurrent reductions being performed. By the end of the reduction, the first thread of the block will place the block's minimum distance into D[blockIdx.x]. While this is happening, one thread per block is also incrementing a device variable using the atomicInc function so that the final thread to call this can aggregate the final

results. It does this by linearly iterating through the D array and placing the final minimum in D[0] before it is copied back to the host in the main function.



Once n reaches 2048, the GPU's performance is now faster than the CPU's performance.


Host to Device and Device to Host Transfer Time vs. k
Transfer Time (ms)
0.3
0.25
0.2
0.15
0.1
0.05
0
4
6
8
10
12
14
16
k
Host to Device
Device to Host