CS 5500 Homework 5

Sally Devitry (A01980316)

Approach

In this assignment, we were tasked with writing a bitonic integer sort using a list of numbers. My program gives each process a random number to sort in a list with the same size as the number of processes.
My program uses process 0 as the master and the rest of the processes as workers. All processes start with the unsorted list. We assume a power of two number of processes. Each two-element segment in the list is a bitonic list. Then, bitonic merges are used to build bigger bitonic lists until we have a sorted list.

Implementation

My program defines 2 important functions, compareUp, and compareDown. The compareUp function does a bitonic sequence sort to an ascending list. The compareDown function does a bitonic sequence sort to a decending list. To determine which way to swap was the tricky part. Depending on the current outer step, the jth bit determines which way they should be swapped (the code for this is as follows).

```
//decide to trade with high or low
for (i = 0; i < dimensions; i++) {
    for (j = i; j >= 0; j--) {
        if (((rank >> (i + 1)) % 2 == 0 && (rank >> j) % 2 == 0)
            || ((rank >> (i + 1)) % 2 != 0 && (rank >> j) % 2 != 0)) {
            compareDown(j, myArray);
        } else {
            compareUp(j, myArray);
        }
    }
}
```

Once all processes have looped through their outer and inner steps, (an MPI barrier is done to ensure they all complete), process 0 prints out the final sorted list.

Some example output:

For 4 processes - `mpirun -np 4 --oversubscribe a.out`

```
sorted array:  0 2 3 3
```

For 8 processes - `mpirun -np 8 --oversubscribe a.out`

```
sorted array:  0 0 1 2 3 7 7 12
```

For 16 processes - `mpirun -np 16 --oversubscribe a.out`

```
sorted array:  0 1 2 2 2 3 3 5 6 7 9 11 11 12 12 32708
```

Concluding Remarks

Overall, understanding the intricacies of how the bitonic sort works was very time consuming. It was a very interesting and new concept to me. I would be interested to see how it performs against other sorts.