Project 3 - Group 5

Team Members

Group Member	R#
Michael Beebe	R11772231
Diego Salas Noain	R11794236
Bandar Alkhalil	R11836831
Yongjian Zhao	R11915830
Denish Otieno	R11743138
Shiva Kumar Neekishetty	R11842757

Required Software

- MPI implementation (we are using Open MPI)
- C Compiler (such as gcc or clang)
- Make
- Bash

Instructions

Compile

To change the MPI wrapper to something other than `mpicc` (such as `mpich`), edit line 1 of the Makefile.

```
make
```

Run

```
./run.sh <desired number of processes>
```

The default number of processes is 4. You can change this by passing a command line argument when executing `run.sh`.

If you get an error saying "permission denied", run

```
chmod +x run.sh
```

then rerun `./run.sh`

Clean Build

```
make clean
```

Code Breakdown

main():

- Initialize distance and edge array, to simply the validation process, we assume all nodes are placed on a straight line, each node is 1 unit apart. You may use other edge values instead.

HW3():

- The parallelization is achieved when each process handles a specific non-overlapping range of the nodes, for `n` nodes with `process_count` processors, each node will handle a size of `chunk` nodes which equals `n` divided by `process_count`, for a specific process with rank `i`, the data range is `[rank * chunk, (rank + 1) * chunk)`.
- Each process will find its local min value and position, then call MPI_Allgather() which works as follows:
    - Given a set of elements distributed across all processes, MPI_Allgather will gather all of the elements to all the processes,
- Each process will then have all the local min values and positions from all other processes (`local_min_val_group` and `local_min_pos_group`), then each process can calculate the global min value and position.
- Finally, we use MPI_gather to combine the `distance` result on process `0`.