Building

This project uses `make` as its build system with the `g++` compiler from the GCC. There are no external dependencies, but you will need a reasonably up to date version of GCC as this project relies on the `C++20` standard. There are several recipes within the `Makefile`, which you can invoke via `make [recipe]` from the command line:

1. `main`, which can be invoked via `make main` or just `make`, creates an unoptimized, generic `main` executable.
2. `debug` creates a `main` binary with debugging symbols, for use with GDB.
3. `profile` creates a `main` binary with profiling information.
4. `optimized` will create an optimized binary for your specific computer:
    1. `-march=native` will use instructions specific to your CPU allowing the use of AVX2, AVX-512, and others which can drastically improve performance
    2. `-O3` will instruct GCC to apply aggressive optimizations.
    3. `-ffast-math` will use fuzzier, less accurate pathways for floating point arithmetic, which for the purposes of this project are a non-issue and boost speed tremendously.
    4. `-flto` will employ Link Time Optimization, to which optimizations at the linking stage of compiling can determine unused functions.
5. `pgo` will apply Profile Guided Optimization, in which an intermediary version of the program with profiling support is created, and a profile is generated by running a common workload. This profile can then be used to determine statistics about program execution, chiefly how many times conditional statements like `if` and `for` are evaluated to true and false, alongside allowing for conditional loop unrolling, to which a loop is “unrolled” into a set of explicit instructions for each “iteration”, which can then be vectorized and run in parallel.