Timothy Lee

CS5345 Advanced Applications Programming

Dr. Corey Clark

Report – Lab 1

**Initial environment setup**

To set up the Emscripten environment, I referred to the official Emscripten website and their documentation at [*https://emscripten.org/docs/getting\_started*](https://emscripten.org/docs/getting_started). First, I had to clone a git repository in order to download the core Emscripten SDK written as a python script.

**Text

Description automatically generated**

Next, upon navigating to the /emsdk directory, I had to do a *git pull* in order to fetch the latest version of the emsdk which was already up-to-date.

Text

Description automatically generated

Then, I had to install and activate the emsdk tools. When installing the emsdk tools using the command, *./emsdk install latest*, I initially ran into the following error.

Text

Description automatically generated

After some research, I came to the conclusion that this was a python issue with it being unable to pass certs related verification (source*:* [*https://github.com/emscripten-core/emscripten/issues/6723*](https://github.com/emscripten-core/emscripten/issues/6723)). I found that installing the certifi package through pip resolved the issue. With certifi installed, I was then able to properly install and activate the appropriate emsdk tools. Finally, I had to activate PATH and other environmental variables in the terminal using the following command.

Text

Description automatically generated

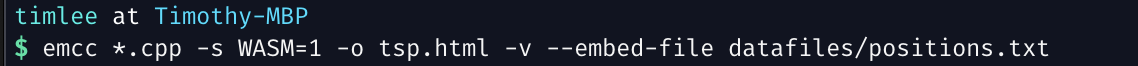
With this command, I would then be able to run emcc anywhere on my system.

To verify compilation and instantiation of my code into web assembly, I wrote a simple helloWorld program in c and compiled it with the following command.



This command uses emcc to compile the c program into web assembly language and also creates an html output so as to easily verify the correct compilation of the code. Upon confirming everything to be working as intended, I moved on to my target codebase.

I decided to use my c++ codebase for travelling salesman algorithm with dynamic programming from CS3353 as my codebase. This codebase takes in an input file as a .txt and uses 20 cities. I had issues compiling this codebase with emcc and the reason turned out to be the process in which Emscripten interacts with input or output files. To load the input file, I had to embed the file during emcc compilation with the following flag.



**Data Comparison and Report**

Timing of the execution of the native and WASM based codebases were done using the *chrono* package with 20 nodes as inputs for the travelling salesman algorithm with dynamic programming. 20 Execution times were logged for both codebases and is displayed in tabular form below.

Table

Description automatically generated

In order to better understand the data, some basic statistical calculations were performed and are displayed below.

Table

Description automatically generated

On average, the native codebase executed about 5.5 times faster than the WASM based codebase compiled on emcc. The WASM based codebase also had a much higher standard deviation of being 21 times higher than the native codebase. The confidence interval was calculated using the standard error. The bar plot below visually illustrates the difference in execution time as well as their variability over 20 executions between the two codebases at 95% confidence.

Chart, box and whisker chart

Description automatically generated

This shows a statistically significant result as even though the 95% confidence interval for the emcc execution time had a much bigger range than one for gcc, the significant level of disparity between the two execution times remains relevant. It is therefore concluded that the execution time of the WASM based codebase runs significantly slower than one done natively by around 5 times.