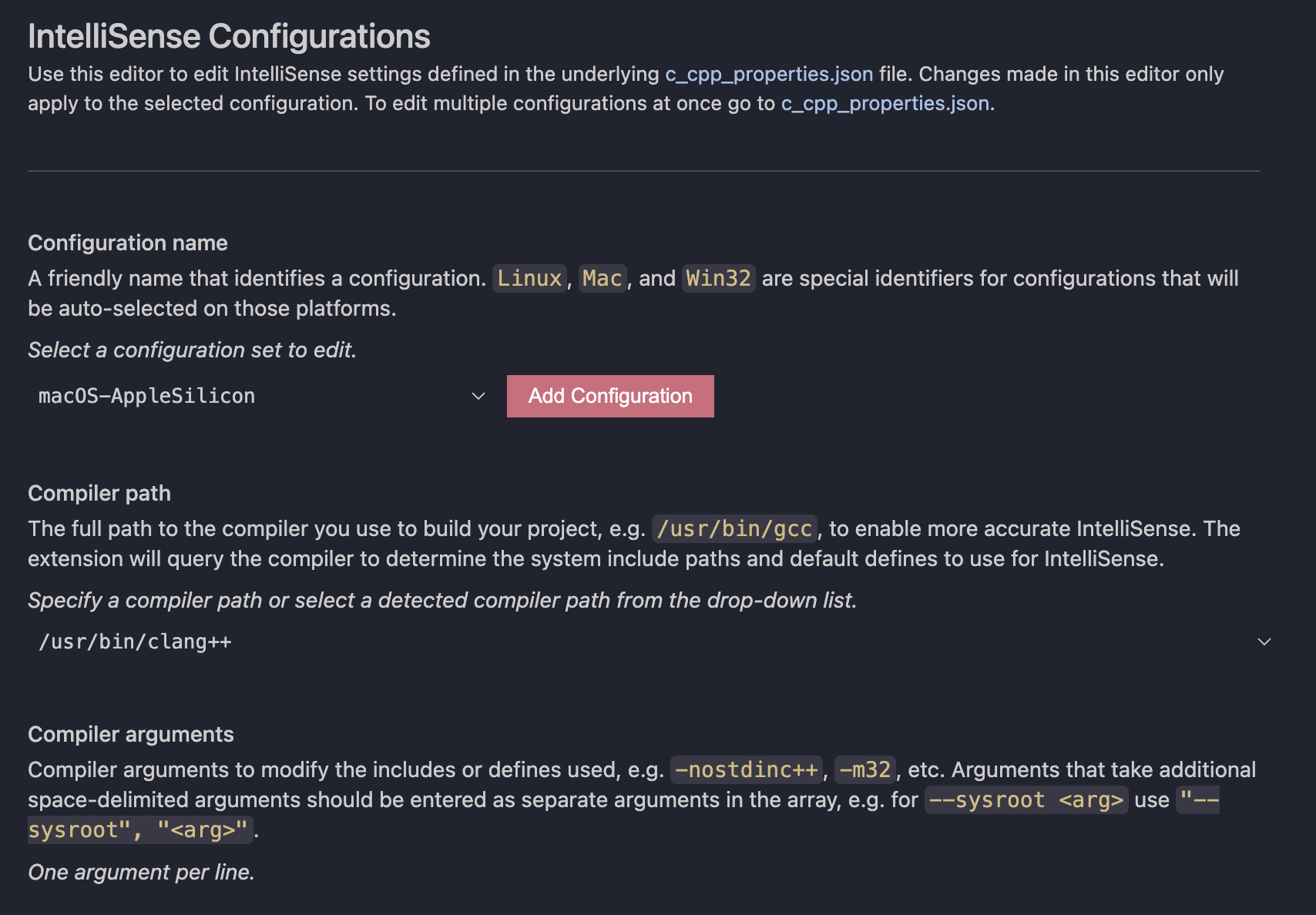
Algorithms Project - Notes and How I worked

Notes:

I worked with cli to comply and installed some outside compiler besides the vscode compiler due to there being a problem for some reason.. and after hours of stackoverflow still no solution was found- BUT I managed to make it work!

Photos for reference..

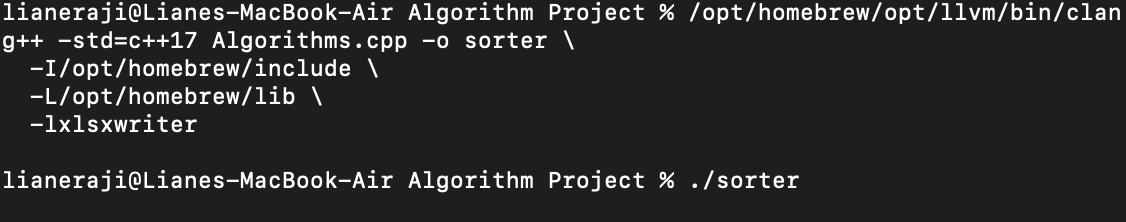
Heres me tweaking VSCode to make compilation work.. it wouldn’t find the librarries and it didn’t





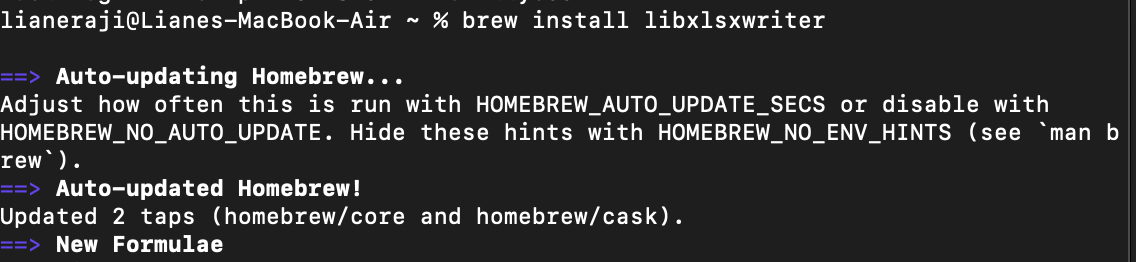
Stack overflow suggested many things in this file (vscode/c\_cpp\_properties.json) such as fetching the include path by writing stuff in terminal gcc -v -E -x c++ - but my solution worked fine for me and these didn’t (<https://stackoverflow.com/questions/65421161/visual-studio-code-cannot-open-source-file-iostream> )

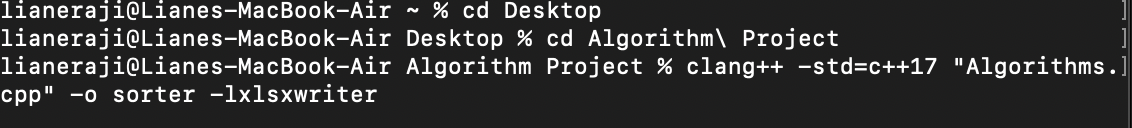
So this is how I ended up compiling the code.. every single time…



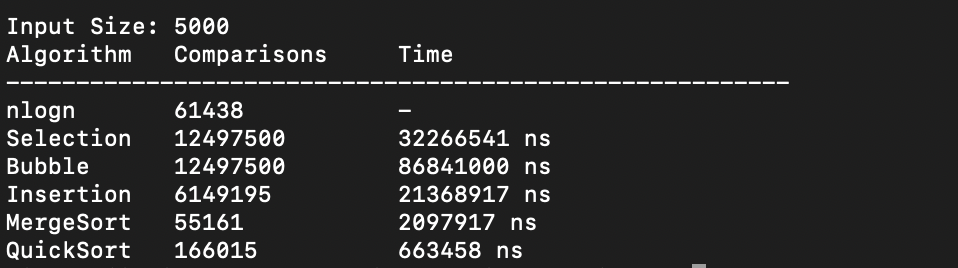
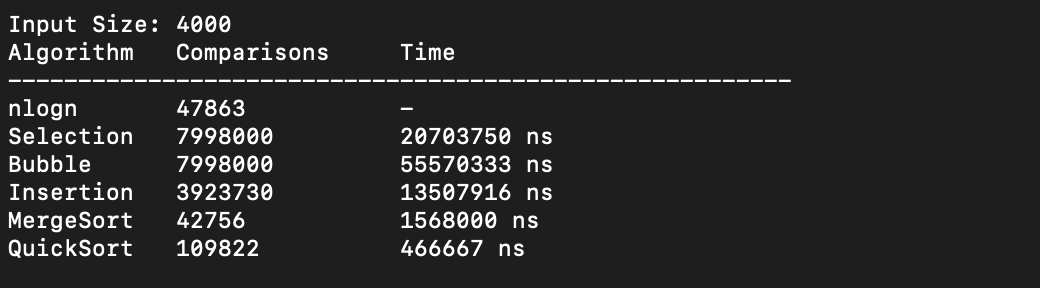
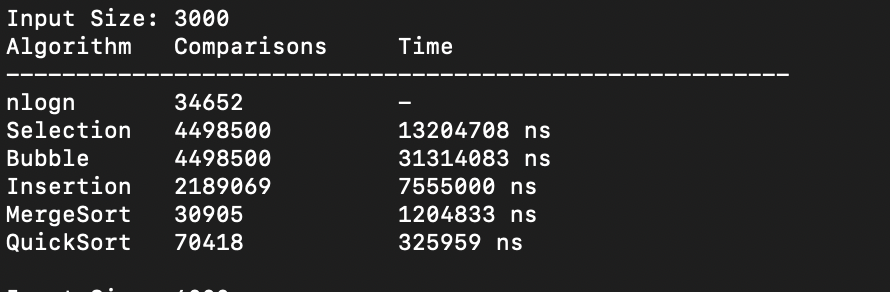
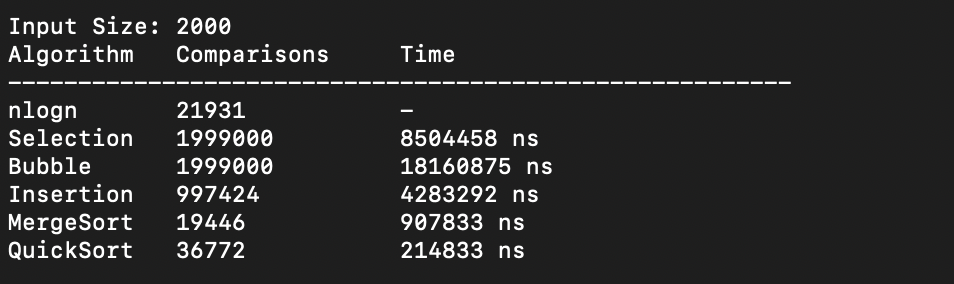
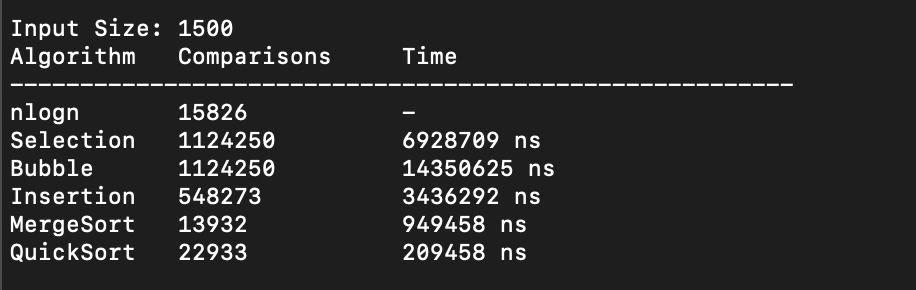
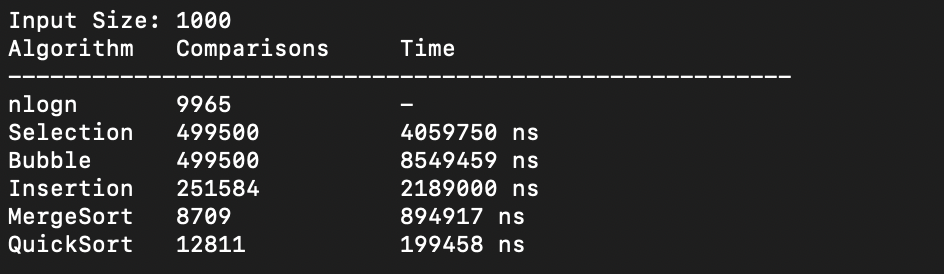
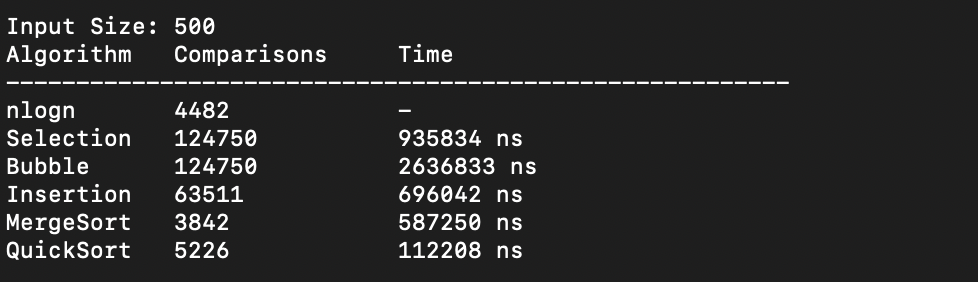
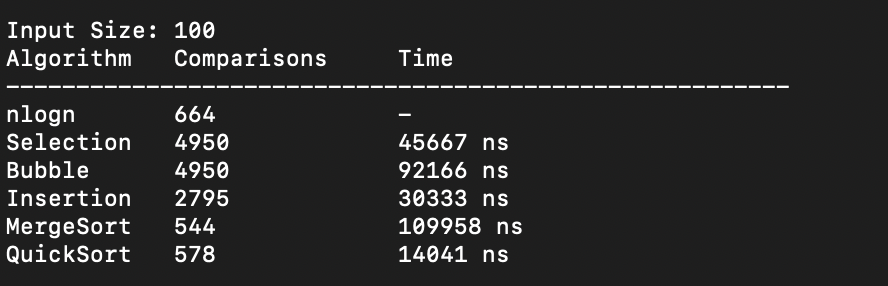
I also wanted to be a bit extra with my project so instead od manually entering values of the code output into a table then making a chart…

I spent hours… HOURS to make it generate time time as a chart in cli as well as it generating a new excel file (or overwriting an existing one with same name) and it generated the graph as well…. Its way harder than it looks even if its like 200-300 lines of code… I had to use this:





To make the output this:

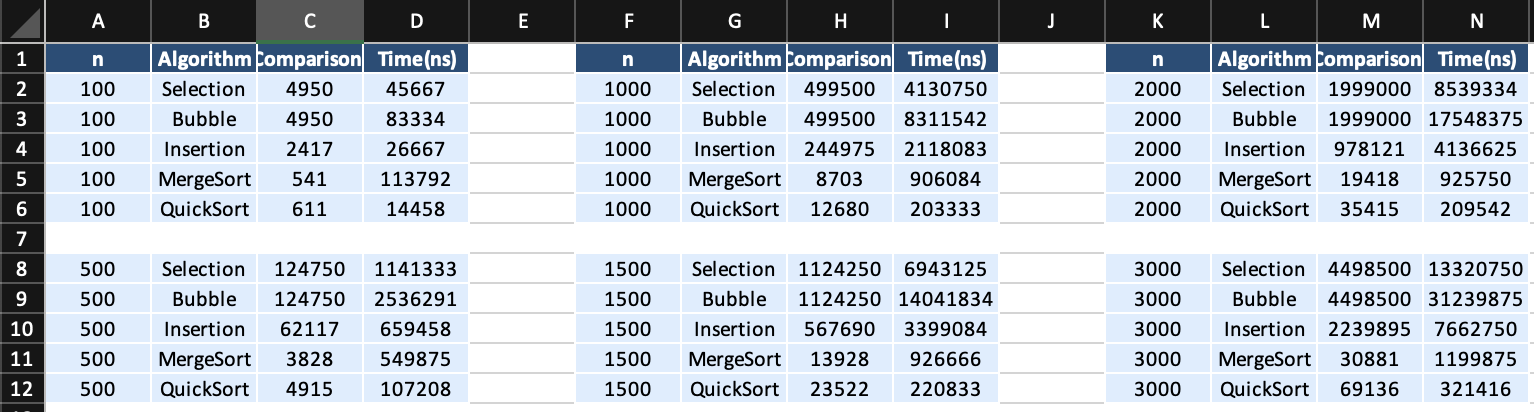


So now I have sorter which is what generates the sorting… as u can see in my first screenshot it was added in the code for compilation since its like a script to run

Heres the excel file updating itself per generation



The output looks like this so far but depending on how much I play with it before I turn it in.. it might look different.. but I made the tables look so tidy and pretty! <3



This code works perfectly as it was used to make the pretty blue table and the neet cli but I had to tweak it because I needed to make it generate graph ☹

#include <iostream>

#include <vector>

#include <algorithm>

#include <chrono>

#include <cstdlib>

#include <ctime>

#include <cmath>

#include <string>

#include <iomanip>

#include <fstream>

#include <sys/stat.h>

#include <xlsxwriter.h>

using *namespace* std;

using *namespace* chrono;

*long* *long* comparisons;

lxw\_workbook \*workbook = nullptr;

lxw\_worksheet \*worksheet = nullptr;

*int* current\_row = 1;

*int* current\_col = 0;

*int* block\_counter = 0;

lxw\_format \*header\_format = nullptr;

lxw\_format \*row\_format = nullptr;

lxw\_format \*separator\_format = nullptr;

*bool* generate\_excel = true;

vector<*long* *long*> selection\_comparisons, bubble\_comparisons, insertion\_comparisons, merge\_comparisons, quick\_comparisons;

vector<*long* *long*> selection\_times, bubble\_times, insertion\_times, merge\_times, quick\_times;

vector<*int*> input\_sizes;

*void* selectionSort(vector<*int*> *arr*) {

comparisons = 0;

*int* n = arr.size();

for (*int* i = 0; i < n - 1; i++) {

*int* min\_idx = i;

for (*int* j = i + 1; j < n; j++) {

comparisons++;

if (arr[j] < arr[min\_idx])

min\_idx = j;

}

swap(arr[i], arr[min\_idx]);

}

}

*void* bubbleSort(vector<*int*> *arr*) {

comparisons = 0;

*int* n = arr.size();

for (*int* i = 0; i < n - 1; i++) {

for (*int* j = 0; j < n - i - 1; j++) {

comparisons++;

if (arr[j] > arr[j + 1])

swap(arr[j], arr[j + 1]);

}

}

}

*void* insertionSort(vector<*int*> *arr*) {

comparisons = 0;

*int* n = arr.size();

for (*int* i = 1; i < n; i++) {

*int* key = arr[i];

*int* j = i - 1;

while (j >= 0) {

comparisons++;

if (arr[j] > key) {

arr[j + 1] = arr[j];

j--;

} else {

break;

}

}

arr[j + 1] = key;

}

}

*long* *long* merge(vector<*int*>& *arr*, *int* *l*, *int* *m*, *int* *r*) {

*long* *long* comps = 0;

*int* n1 = m - l + 1;

*int* n2 = r - m;

vector<*int*> L(n1), R(n2);

for (*int* i = 0; i < n1; i++) L[i] = arr[l + i];

for (*int* j = 0; j < n2; j++) R[j] = arr[m + 1 + j];

*int* i = 0, j = 0, k = l;

while (i < n1 && j < n2) {

comps++;

if (L[i] <= R[j]) arr[k++] = L[i++];

else arr[k++] = R[j++];

}

while (i < n1) arr[k++] = L[i++];

while (j < n2) arr[k++] = R[j++];

return comps;

}

*long* *long* mergeSortRec(vector<*int*>& *arr*, *int* *l*, *int* *r*) {

*long* *long* comps = 0;

if (l < r) {

*int* m = l + (r - l) / 2;

comps += mergeSortRec(arr, l, m);

comps += mergeSortRec(arr, m + 1, r);

comps += merge(arr, l, m, r);

}

return comps;

}

*void* mergeSort(vector<*int*> *arr*) {

comparisons = mergeSortRec(arr, 0, arr.size() - 1);

}

*long* *long* quickSortRec(vector<*int*>& *arr*, *int* *low*, *int* *high*) {

*long* *long* comps = 0;

if (low < high) {

*int* pivot = arr[high];

*int* i = (low - 1);

for (*int* j = low; j < high; j++) {

comps++;

if (arr[j] < pivot) {

i++;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

*int* pi = i + 1;

comps += quickSortRec(arr, low, pi - 1);

comps += quickSortRec(arr, pi + 1, high);

}

return comps;

}

*void* quickSort(vector<*int*> *arr*) {

comparisons = quickSortRec(arr, 0, arr.size() - 1);

}

*void* writeExcelHeader(*int* *row*, *int* *col\_offset*) {

worksheet\_write\_string(worksheet, row, col\_offset + 0, "n", header\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 1, "Algorithm", header\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 2, "Comparisons", header\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 3, "Time(ns)", header\_format);

}

*void* writeExcelRow(*int* *row*, *int* *col\_offset*, *int* *n*, const string& *algo*, *long* *long* *comps*, *long* *long* *time\_ns*) {

worksheet\_write\_number(worksheet, row, col\_offset + 0, n, row\_format);

worksheet\_write\_string(worksheet, row, col\_offset + 1, algo.c\_str(), row\_format);

worksheet\_write\_number(worksheet, row, col\_offset + 2, comps, row\_format);

worksheet\_write\_number(worksheet, row, col\_offset + 3, time\_ns, row\_format);

}

*void* writeExcelSeparatorRow(*int* *row*, *int* *col\_offset*) {

for (*int* i = 0; i < 4; ++i)

worksheet\_write\_blank(worksheet, row, col\_offset + i, separator\_format);

}

*void* runSort(*void*(\**sortFunc*)(vector<*int*>), vector<*int*> *arr*, const string& *name*, *int* *n*, *int* *col\_offset*, vector<*long* *long*>& *comp\_list*, vector<*long* *long*>& *time\_list*) {

vector<*int*> copy = arr;

*auto* start = high\_resolution\_clock::now();

sortFunc(copy);

*auto* end = high\_resolution\_clock::now();

*auto* duration = duration\_cast<nanoseconds>(end - start).count();

cout << left << setw(12) << name << setw(16) << comparisons << duration << " ns" << endl;

if (generate\_excel) {

writeExcelRow(current\_row, col\_offset, n, name, comparisons, duration);

current\_row++;

}

comp\_list.push\_back(comparisons);

time\_list.push\_back(duration);

}

*int* main() {

workbook = workbook\_new("projectexcel.xlsx");

worksheet = workbook\_add\_worksheet(workbook, NULL);

header\_format = workbook\_add\_format(workbook);

format\_set\_bold(header\_format);

format\_set\_font\_color(header\_format, LXW\_COLOR\_WHITE);

format\_set\_bg\_color(header\_format, 0x1F4E78);

format\_set\_align(header\_format, LXW\_ALIGN\_CENTER);

format\_set\_border(header\_format, LXW\_BORDER\_THIN);

format\_set\_border\_color(header\_format, LXW\_COLOR\_WHITE);

row\_format = workbook\_add\_format(workbook);

format\_set\_bg\_color(row\_format, 0xDDEEFF);

format\_set\_align(row\_format, LXW\_ALIGN\_CENTER);

format\_set\_border(row\_format, LXW\_BORDER\_THIN);

format\_set\_border\_color(row\_format, LXW\_COLOR\_WHITE);

separator\_format = workbook\_add\_format(workbook);

format\_set\_bg\_color(separator\_format, LXW\_COLOR\_WHITE);

srand(time(0));

vector<*int*> sizes = {100, 500, 1000, 1500, 2000, 3000, 4000, 5000};

for (*int* n : sizes) {

input\_sizes.push\_back(n);

if (block\_counter == 2) {

writeExcelSeparatorRow(current\_row, current\_col);

current\_row = 1;

current\_col += 5;

block\_counter = 0;

}

*long* *long* theo\_nlogn = static\_cast<*long* *long*>(n \* log2(n));

cout << "\nInput Size: " << n << endl;

cout << left << setw(12) << "Algorithm" << setw(16) << "Comparisons" << "Time" << endl;

cout << "--------------------------------------------------------" << endl;

cout << left << setw(12) << "nlogn" << setw(16) << theo\_nlogn << "-" << endl;

if (current\_row == 1)

writeExcelHeader(0, current\_col);

writeExcelRow(current\_row, current\_col, n, "nlogn", theo\_nlogn, 0);

current\_row++;

vector<*int*> base(n);

generate(base.begin(), base.end(), []() { return rand() % 101; });

runSort(selectionSort, base, "Selection", n, current\_col, selection\_comparisons, selection\_times);

runSort(bubbleSort, base, "Bubble", n, current\_col, bubble\_comparisons, bubble\_times);

runSort(insertionSort, base, "Insertion", n, current\_col, insertion\_comparisons, insertion\_times);

runSort(mergeSort, base, "MergeSort", n, current\_col, merge\_comparisons, merge\_times);

runSort(quickSort, base, "QuickSort", n, current\_col, quick\_comparisons, quick\_times);

block\_counter++;

if (n != sizes.back()) {

writeExcelSeparatorRow(current\_row, current\_col);

current\_row++;

}

}

workbook\_close(workbook);

return 0;

}

Trying to add graph is a bit difficult but I will add more info on my official word doc.. not these notes ☺