

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

IntelliSense Configurations

Use this editor to edit IntelliSense settings defined in the underlying `c_cpp_properties.json` file. Changes made in this editor only apply to the selected configuration. To edit multiple configurations at once go to `c_cpp_properties.json`.

Configuration name

A friendly name that identifies a configuration. `Linux`, `Mac`, and `Win32` are special identifiers for configurations that will be auto-selected on those platforms.

Select a configuration set to edit.

macOS–AppleSilicon



Add Configuration

Compiler path

The full path to the compiler you use to build your project, e.g. `/usr/bin/gcc`, to enable more accurate IntelliSense. The extension will query the compiler to determine the system include paths and default defines to use for IntelliSense.

Specify a compiler path or select a detected compiler path from the drop-down list.

/usr/bin/clang++



Compiler arguments

Compiler arguments to modify the includes or defines used, e.g. `-nostdinc++`, `-m32`, etc. Arguments that take additional space-delimited arguments should be entered as separate arguments in the array, e.g. for `--sysroot <arg>` use `"--sysroot", "<arg>"`.

One argument per line.

```

1  {
2      "configurations": [
3      {
4          "name": "Mac",
5          "includePath": [
6              "/Library/Developer/CommandLineTools/usr/include/c++/v1",
7              "/Library/Developer/CommandLineTools/usr/lib/clang/16/include",
8              "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include",
9              "/Library/Developer/CommandLineTools/usr/include"
10         ],
11         "macFrameworkPath": [
12             "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/System/Library/Frameworks",
13             "/System/Library/Frameworks",
14             "/Library/Frameworks"
15         ],
16         "compilerPath": "/usr/bin/clang++",
17         "cStandard": "c11",
18         "cppStandard": "c++17",
19         "intelliSenseMode": "macos-clang-arm64"
20     }
21 ],
22 "version": 4
23 }

```

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...

```

lianeraji@Lianes-MacBook-Air Algorithm Project % /opt/homebrew/opt/llvm/bin/clang++ -std=c++17 Algorithms.cpp -o sorter \
-I/opt/homebrew/include \
-L/opt/homebrew/lib \
-lxlsxwriter

lianeraji@Lianes-MacBook-Air Algorithm Project % ./sorter

```

I also wanted to be a bit extra with my project so instead of 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:

```

lianeraji@Lianes-MacBook-Air ~ % brew install libxlsxwriter

==> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man b
rew`).
==> Auto-updated Homebrew!
Updated 2 taps (homebrew/core and homebrew/cask).
==> New Formulae

lianeraji@Lianes-MacBook-Air ~ % cd Desktop
lianeraji@Lianes-MacBook-Air Desktop % cd Algorithm\ Project
lianeraji@Lianes-MacBook-Air Algorithm Project % clang++ -std=c++17 "Algorithms.
cpp" -o sorter -lsxwriter

```

To make the output this:

```

Input Size: 100
Algorithm    Comparisons    Time
-----
nlogn        664            -
Selection    4950           45667 ns
Bubble       4950           92166 ns
Insertion    2795           30333 ns
MergeSort    544            109958 ns
QuickSort    578            14041 ns

Input Size: 500
Algorithm    Comparisons    Time
-----
nlogn        4482           -
Selection    124750         935834 ns
Bubble       124750         2636833 ns
Insertion    63511          696042 ns
MergeSort    3842           587250 ns
QuickSort    5226           112208 ns

Input Size: 1000
Algorithm    Comparisons    Time
-----
nlogn        9965           -
Selection    499500         4059750 ns
Bubble       499500         8549459 ns
Insertion    251584         2189000 ns
MergeSort    8709           894917 ns
QuickSort    12811          199458 ns

```

Input Size: 1500

Algorithm	Comparisons	Time
-----------	-------------	------

nlogn	15826	-
Selection	1124250	6928709 ns
Bubble	1124250	14350625 ns
Insertion	548273	3436292 ns
MergeSort	13932	949458 ns
QuickSort	22933	209458 ns

Input Size: 2000

Algorithm	Comparisons	Time
-----------	-------------	------

nlogn	21931	-
Selection	1999000	8504458 ns
Bubble	1999000	18160875 ns
Insertion	997424	4283292 ns
MergeSort	19446	907833 ns
QuickSort	36772	214833 ns

Input Size: 3000

Algorithm	Comparisons	Time
-----------	-------------	------

nlogn	34652	-
Selection	4498500	13204708 ns
Bubble	4498500	31314083 ns
Insertion	2189069	7555000 ns
MergeSort	30905	1204833 ns
QuickSort	70418	325959 ns

Input Size: 4000		
Algorithm	Comparisons	Time

nlogn	47863	—
Selection	7998000	20703750 ns
Bubble	7998000	55570333 ns
Insertion	3923730	13507916 ns
MergeSort	42756	1568000 ns
QuickSort	109822	466667 ns
Input Size: 5000		
Algorithm	Comparisons	Time

nlogn	61438	—
Selection	12497500	32266541 ns
Bubble	12497500	86841000 ns
Insertion	6149195	21368917 ns
MergeSort	55161	2097917 ns
QuickSort	166015	663458 ns

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

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	n	Algorithm	Comparison	Time(ns)		n	Algorithm	Comparison	Time(ns)		n	Algorithm	Comparison	Time(ns)
2	100	Selection	4950	45667		1000	Selection	499500	4130750		2000	Selection	1999000	8539334
3	100	Bubble	4950	83334		1000	Bubble	499500	8311542		2000	Bubble	1999000	17548375
4	100	Insertion	2417	26667		1000	Insertion	244975	2118083		2000	Insertion	978121	4136625
5	100	MergeSort	541	113792		1000	MergeSort	8703	906084		2000	MergeSort	19418	925750
6	100	QuickSort	611	14458		1000	QuickSort	12680	203333		2000	QuickSort	35415	209542
7														
8	500	Selection	124750	1141333		1500	Selection	1124250	6943125		3000	Selection	4498500	13320750
9	500	Bubble	124750	2536291		1500	Bubble	1124250	14041834		3000	Bubble	4498500	31239875
10	500	Insertion	62117	659458		1500	Insertion	567690	3399084		3000	Insertion	2239895	7662750
11	500	MergeSort	3828	549875		1500	MergeSort	13928	926666		3000	MergeSort	30881	1199875
12	500	QuickSort	4915	107208		1500	QuickSort	23522	220833		3000	QuickSort	69136	321416

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--;
            }
        }
        arr[j + 1] = key;
    }
}

```

```

        } 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 😊