CptS 223 – Advanced Data Structures in C++

Micro-assignment 4: Parallel Programming and Heaps

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

Implement solutions to **parallel algorithms (openmp)**
Implement solutions to **percolate down** and **percolate up** algorithms for binary heaps

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

1. Analyze a basic set of requirements and apply parallel design principles for a problem
2. Describe and analyze binary heaps
3. Edit, build, and run programs through a WSL platform, Linux environment, or MacOS system

III. Overview & Requirements:

For this micro assignment, you should start with the code template provided on Canvas. You are required to implement the following functions in OpenMP.h and Heap.h. After that, run test cases in main.cpp (should not need to write anything in main.cpp).

1. <u>OpenMP.h</u>

(50 pts) calc_max(). Find the maximum value in the data array using OpenMP with at least two threads.

The beginning and ending of this function are provided in the template. You just need to implement the logic between the “start your code here” and “stop here”. Do NOT change other parts of the function.

*Two examples: cal_sum() and hello_world():
They are provided in OpenMP.h. The purpose of these two functions are to help you understand how OpenMP works. In main.cpp, runOpenMP() calls hello_world() and cal_sum(), where you may learn OpenMP.

For finishing cal_max(), you may follow the two examples to implement your code in cal_max().

2. Heap.h: percolateDown and percolateUp

A min-heap (the smallest item at the top). The first item in the array (std::vector) is a placeholder as we mentioned in the class. The core logic of the two functions mentioned below could be copied from our slides. But you may need to adjust it a little bit to fit into the template. Do NOT change other parts of the Heap.h.

(25 pts) `percolateDown`
(25 pts) `percolateUp`

(Hint: check Textbook, Page 249 to 253, Chapter 6.3.3, or Lecture slides “8_heaps_1.pdf”, page 15 to 30, for both operations.)

These functions cause the items at the supplied locations to "percolate down" and “percolate up” the heap until the min-heap property is satisfied.

The `percolateDown()` method is called in `pop()` operation; check the `pop()` function inside `Heap` class to see how `percolateDown()` is called.

The `percolateUp()` function is called on `push()` calls. It adds an item to the end of the heap and move it until it satisfies the heap properties of our min heap.

3. Test cases:

Main.cpp contains a number of test cases. Do NOT change the main.cpp.

1) The cal_max() will print the max value found by stl and OpenMP. If your implementation is correct, the two values should match.
2) The runHeap() function will push a number of random values into the heap and pop the values one by one. It compares the popped values with the values sorted by stl. If your implementation is correct, the assertion function used in the loop in main.cpp should pass. Currently, they are failed.

4. Instruction for CMake verification:

Successfully building your project using CMake: save all generated files and executables.
Hint: CMakeLists.txt provided for both WSL and MacOS systems.

4.1. If you are using WSL or other platforms:

Copy “CMakeLists_wsl.txt” to a new file “CMakeLists.txt” (simply “cp CMakeLists_wsl.txt CMakeLists.txt” in terminal) and use it for building your project.

See below the building process and running executable.

```
yan@DESKTOP-HSN7G8B: /mnt/c/Users/yanya/Downloads/start_code_test/MA4
(base) yan@DESKTOP-HSN7G8B:/mnt/c/Users/yanya/Downloads/start_code_test/MA4$ mkdir build
(base) yan@DESKTOP-HSN7G8B:/mnt/c/Users/yanya/Downloads/start_code_test/MA4$ cmake -S . -B build
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/c/Users/yanya/Downloads/start_code_test/MA4/build
(base) yan@DESKTOP-HSN7G8B:/mnt/c/Users/yanya/Downloads/start_code_test/MA4$ cmake --build build
Scanning dependencies of target CPTS_223_MA4-executable
[ 50%] Building CXX object CMakeFiles/CPTS_223_MA4-executable.dir/main.cpp.o
In file included from /mnt/c/Users/yanya/Downloads/start_code_test/MA4/main.cpp:2:
/mnt/c/Users/yanya/Downloads/start_code_test/MA4/Heap.h: In instantiation of ævoid Heap<T>::percol
ateDown(unsigned int) [with T = int]Æ:
/mnt/c/Users/yanya/Downloads/start_code_test/MA4/Heap.h:129:4:   required from æT Heap<T>::pop() [
with T = int]Æ
/mnt/c/Users/yanya/Downloads/start_code_test/MA4/main.cpp:29:9:   required from here
/mnt/c/Users/yanya/Downloads/start_code_test/MA4/Heap.h:40:18: warning: comparison of integer expr
essions of different signedness: æintÆ and æstd::vector<int>::size_typeÆ (aka ælong unsigned intÆ)
 [-Wsign-compare]
   40 |     if (leftchild > _items.size() - 1) // hole is leaf node
      |         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/mnt/c/Users/yanya/Downloads/start_code_test/MA4/Heap.h:47:18: warning: comparison of integer expr
essions of different signedness: æintÆ and æstd::vector<int>::size_typeÆ (aka ælong unsigned intÆ)
 [-Wsign-compare]
   47 |     if (leftchild == _items.size() - 1 || _items[leftchild] < _items[rightchild])
      |         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[100%] Linking CXX executable CPTS_223_MA4-executable
[100%] Built target CPTS_223_MA4-executable
(base) yan@DESKTOP-HSN7G8B:/mnt/c/Users/yanya/Downloads/start_code_test/MA4$ build/CPTS_223_MA4-ex
ecutable

Start hello_world()
Welcome from thread = 14
Welcome from thread = 2
Welcome from thread = 0
Number of threads = 16: this message from thread #0
Welcome from thread = 10
Welcome from thread = 12
Welcome from thread = 3
Welcome from thread = 15
Welcome from thread = 7
Welcome from thread = 13
Welcome from thread = 4
Welcome from thread = 11
Welcome from thread = 8
Welcome from thread = 1
Welcome from thread = 5
Welcome from thread = 9
Welcome from thread = 6

Finish hello_world()

Start cal_sum()
Elapsed time = 75863[microseconds]

Finish cal_sum(): assert pass!

Start cal_max()
Elapsed time = 49482[microseconds]
Max found by stl: 99999999, max found by OpenMP 99999999

Finish cal_max()

Start runHeap()

Start runHeap(): assert pass!
(base) yan@DESKTOP-HSN7G8B:/mnt/c/Users/yanya/Downloads/start_code_test/MA4$
```

4.2. If you are using MacOS:

There are several steps to enable MacOS to support OpenMP (Linux or WSL support OpenMP automatically).

4.2.1. Install OpenMP:

“brew install libomp”

Check this link for more details: https://formulae.brew.sh/formula/libomp

if you have the following errors:

```
(base) yanyan@wless-user-172019033081 MA4 % brew install libomp
Error: Cannot install under Rosetta 2 in ARM default prefix (/opt/homebrew)!
To rerun under ARM use:
    arch -arm64 brew install ...
To install under x86_64, install Homebrew into /usr/local.
(base) yanyan@wless-user-172019033081 MA4 %
```

it means that your laptop is built under ARM architecture, rather than x86_64 architecture.

In this case, use “arch -arm64 brew install” instead for installation, as follows

```
(base) yanyan@wless-user-172019033081 MA4 % brew install libomp
Error: Cannot install under Rosetta 2 in ARM default prefix (/opt/homebrew)!
To rerun under ARM use:
    arch -arm64 brew install ...
To install under x86_64, install Homebrew into /usr/local.
(base) yanyan@wless-user-172019033081 MA4 % arch -arm64 brew install libomp
==> Downloading https://ghcr.io/v2/homebrew/core/libomp/manifests/13.0.0
Already downloaded: /Users/yanyan/Library/Caches/Homebrew/downloads/e5d1162de2115
2d4e8edcdcad88daf8290f53769dd8c7677d94025982d5594fc--libomp-13.0.0.bottle_manifes
t.json
==> Downloading https://ghcr.io/v2/homebrew/core/libomp/blobs/sha256:2a7253a4e9ff
Already downloaded: /Users/yanyan/Library/Caches/Homebrew/downloads/23065afe04a75
93ccae727b6178a684abb78c6f38179096dd70ed6ff50e8c4c2--libomp--13.0.0.arm64_montere
y.bottle.tar.gz
==> Pouring libomp--13.0.0.arm64_monterey.bottle.tar.gz
🍺  /opt/homebrew/Cellar/libomp/13.0.0: 9 files, 1.6MB
==> Running `brew cleanup libomp`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
(base) yanyan@wless-user-172019033081 MA4 %
```

After installation, double check the location for the libomp installation:

“ls /opt/homebrew/opt | grep omp”

If you can see there is a directory named “libomp”, then this installation is successful and also you will be able to access this library in future use.

```
(base) yanyan@wless-user-172019033081 MA4 % brew install libomp
Error: Cannot install under Rosetta 2 in ARM default prefix (/opt/homebrew)!
To rerun under ARM use:
    arch -arm64 brew install ...
To install under x86_64, install Homebrew into /usr/local.
(base) yanyan@wless-user-172019033081 MA4 % arch -arm64 brew install libomp
==> Downloading https://ghcr.io/v2/homebrew/core/libomp/manifests/13.0.0
Already downloaded: /Users/yanyan/Library/Caches/Homebrew/downloads/e5d1162de2115
2d4e8edcdcad88daf8290f53769dd8c7677d94025982d5594fc--libomp-13.0.0.bottle_manifes
t.json
==> Downloading https://ghcr.io/v2/homebrew/core/libomp/blobs/sha256:2a7253a4e9ff
Already downloaded: /Users/yanyan/Library/Caches/Homebrew/downloads/23065afe04a75
93ccae727b6178a684abb78c6f38179096dd70ed6ff50e8c4c2--libomp--13.0.0.arm64_montere
y.bottle.tar.gz
==> Pouring libomp--13.0.0.arm64_monterey.bottle.tar.gz
🍺  /opt/homebrew/Cellar/libomp/13.0.0: 9 files, 1.6MB
==> Running `brew cleanup libomp`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
(base) yanyan@wless-user-172019033081 MA4 % ls /opt/homebrew/opt | grep omp
libomp
(base) yanyan@wless-user-172019033081 MA4 %
```

*Why we need to locate this “libomp” directory?

In CMakeLists_macos.txt, we set up the CMAKE_CXX_FLAGS by:

“set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Xpreprocessor -fopenmp -I/opt/homebrew/opt/libomp/include")”, where “-I/opt/homebrew/opt/libomp/include” is the target directory for using libomp installed by brew.

Below is a screenshot of “CMakeLists_macos.txt”: the highlited line is the one need to locate libomp

CMakeLists_macos.txt

```
cmake_minimum_required(VERSION 3.16)
project(CPTS_223_MA4)

set(CMAKE_CXX_STANDARD 11)

set(CMAKE_CXX_FLAGS  "${CMAKE_CXX_FLAGS} -Wall -Xpreprocessor -fopenmp -I/opt/homebrew/opt/libomp/
include")


add_executable(CPTS_223_MA4 main.cpp)

find_package(OpenMP REQUIRED) # Find the package
target_link_libraries(${PROJECT_NAME} ${OpenMP_CXX_LIBRARIES}) # Link against it for C++
```

4.2.2. Building your project using CMake

Copy “CMakeLists_macos.txt” to a new file “CMakeLists.txt” (simply “cp CMakeLists_macos.txt CMakeLists.txt” in terminal) and use it for building your project.

```
To install under x86_64, install Homebrew into /usr/local.
(base) yanyan@wless-user-172019033081 MA4 % arch -arm64 brew install libomp
==> Downloading https://ghcr.io/v2/homebrew/core/libomp/manifests/13.0.0
Already downloaded: /Users/yanyan/Library/Caches/Homebrew/downloads/e5d1162de2115
2d4e8edcdcad88daf8290f53769dd8c7677d94025982d5594fc--libomp-13.0.0.bottle_manifes
t.json
==> Downloading https://ghcr.io/v2/homebrew/core/libomp/blobs/sha256:2a7253a4e9ff
Already downloaded: /Users/yanyan/Library/Caches/Homebrew/downloads/23065afe04a75
93ccae727b6178a684abb78c6f38179096dd70ed6ff50e8c4c2--libomp--13.0.0.arm64_montere
y.bottle.tar.gz
==> Pouring libomp--13.0.0.arm64_monterey.bottle.tar.gz
🍺  /opt/homebrew/Cellar/libomp/13.0.0: 9 files, 1.6MB
==> Running `brew cleanup libomp`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
(base) yanyan@wless-user-172019033081 MA4 % ls /opt/homebrew/opt | grep omp
libomp
(base) yanyan@wless-user-172019033081 MA4 % cmake -S . -B build
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/yanyan/Dropbox/teaching_at_WSU/2021_f
all/CPTS223/MA4_2021Fall/start_code_test/MA4/build
(base) yanyan@wless-user-172019033081 MA4 % cmake --build build
[ 50%] Building CXX object CMakeFiles/CPTS_223_MA4.dir/main.cpp.o
[100%] Linking CXX executable CPTS_223_MA4
[100%] Built target CPTS_223_MA4
(base) yanyan@wless-user-172019033081 MA4 %
```

4.2.3. Run the executable

```
(base) yanyan@wless-user-172019033081 MA4 % cmake --build build
Consolidate compiler generated dependencies of target CPTS_223_MA4
[ 50%] Building CXX object CMakeFiles/CPTS_223_MA4.dir/main.cpp.o
[100%] Linking CXX executable CPTS_223_MA4
[100%] Built target CPTS_223_MA4
(base) yanyan@wless-user-172019033081 MA4 % build/CPTS_223_MA4

Start hello_world()
Welcome from thread = 1
Welcome from thread = 5
Welcome from thread = 0
Number of threads = 8: this message from thread #0
Welcome from thread = 2
Welcome from thread = 4
Welcome from thread = 6
Welcome from thread = 3
Welcome from thread = 7

Finish hello_world()

Start cal_sum()
Elapsed time = 130593[microseconds]

Finish cal_sum(): assert pass!

Start cal_max()
Elapsed time = 196982[microseconds]
max found by stl: 99999998, max found by OpenMP 99999998

Finish cal_max()

Start runHeap()

Start runHeap(): assert pass!
(base) yanyan@wless-user-172019033081 MA4 %
```

IV. Submitting Assignments:

1. Option 1: Canvas

Zip all source files (including CMakeLists.txt, CMake generated files, etc., if any) and upload it to Canvas.

2. Option 2: Github

1) On your local file system, and inside of your Git repo for the class, create a new branch called MA4, push all files of the given project to the MA4 branch of your private GitHub repo created in PA1.
2) Submission: You must submit a URL link to the branch of your private GitHub repository. Please make sure the instructor and TAs (GitHub account listed in Syllabus) are the collaborators of your repository. Otherwise, we won't be able to see your repository. DO NOT CREATE NEW REPO.
3) Do not push new commits the branch after you submit your URL to Canvas otherwise it might be considered as late submission.

V. Grading Guidelines:

This assignment is worth 100 points.