EC 504 – Spring 2023 – Software Homework 3

Due after midterm:

In Part 1 put the code with makefile in `/projectnb/ec504rb/students/username/HW3`

In Part 2 put plots, fits and description of analysis in subdirectory `/projectnb/ec504rb/students/username/HW3/analysis`**. This will be entirely pdf files.**

You will want to modify the source code in Part 1 to do this analysis but DO NOT turn in this modified code. The Part 1 code is being auto-graded and has to be completed only as required.

Using Max Heap to Sort in Place

In first part this is an exercise is to run finish programing and run at code implementing Max Heap with in place Heap Sort algorithm describe in class AND in detail in **CRLS in Sec 6.4.** The second part is to use it sort random array of different size N and verify this this is yet another $O(N \log N)$ sorting algorithm by plotting and doing a least square fit using error bars.

Part1: Code Developing and Testing

The basic template code is as usual in GitHUB at `HW3_codes/heap.cpp` Be careful to submit the code with a makefile so that is compiles and runs on CCS. It must return a correctly sorted array,

The code builds a Max heap for n elements as and array `int HeapArray[n+1];` of $n+1$ setting elements by placing the integers setting `HeapArray[0] = n` and copying the elements putting the elements in sequence into `HeapArray[i], for i = 1,2,..., n`

Complete the code in `HW3_codes/heap.cpp` to enable:

```
(1) Insert random sequence to Heap array
(2) Bottom up Heapify for Max Heap
(3) Delete any key and restore Max Heap
(4) Insert new key and restore Max Heap
(4) Sort in place and print out array
```

Put final code with `Makefile` in `/projectnb/ec504rb/students/username/HW3`

Part 2: Plotting and Analysis

Analyze the behavior of this code and present plots to support this analysis.You will want to modify the code in Part1 to do nice plots and you can use the source code `makeSingleList.cpp` to generate more input files as needed. Plots are easily done in gnuplot with lot's of example on the web. It takes practice to do nice plots. A few examples are on GitHUB at Plotting_and_Fitting

```
(1) Plot timings for range of sizes n = 8, 16, 32,....2^20
so that you can fit performance to  T(n) = a + b n Log[n] + c n*n
to  T(n) = a + b n Log[n] + c n*n. You can use any fitting
program but it is really easy using gnuplot.

(2)  Improve the fit you can repeat several tunes
(say 10 times) each size n = 8, 16, 32,....2^20 by calling a randomizing
function in the sort.h. Then as described in HW2 you can make a table
of average time for each n and error bars.

(3) Now repeat Step (1) above  for n = 8, 16, 32,....2^20
finding best fit to  T(n) = a + b n Log[n] + c n*n  the averages
within the error bars.
```

Place these figures and fitting parameter with a short explanation in
`/projectnb/ec504rb/students/username/HW3/analysis`