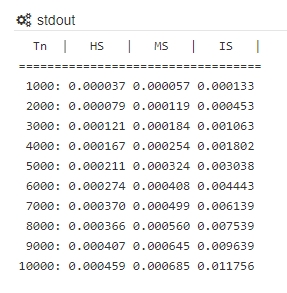
104062261 HW5 Report

Q1. Compare the running time of Heapsort with merge sort and insertion sort.

A1. I run those three sorting algorithms with the same test case, and we can see that the heapsort algorithm is the fastest one. When the input data is much larger, we can see the insertion sort is way slower than the other two algorithms. Since my computer have some trouble to using <time.h>, I’m using the online compiler to finish this question, you can check my code in here: <https://ideone.com/YDNAKK>

Q2. Show and explain the best-case and the worst-case running times for Heapsort.

A2. MAX‐HEAPIFY run in , and BUILD‐MAX‐HEAP run in .

Heapsort takes time since we call the BUILD‐MAX‐HEAP, and each n-1 calls the MAX‐HEAPIFY. The running times of the best-case and the worst-case are the same.

Q3. Why is the following for loop begins from ⌊**A**.length/2⌋?

BUILD-MAX-HEAP(A)

1 **A**.heap-size = **A**.length

2 **for** i = ⌊𝐀.length/2⌋ **downto** 1

3 MAX-HEAPIFY(**A**, i)

A3. Because we are going to compare the non-leaf node and its children, so we don’t have to start at the leaf-node and waste time. Since we know that one non-leaf node can have two children, so using ⌊𝐀.length/2⌋ we can get the last non-leaf node position in the tree.