Problem 2.1. (2pt) Prove $\lg(n!) = \Theta(n \log n)$ when n is an odd number. You can (1) try the similar approach like what we did in the lecture, or (2) find the answer based on the result for even numbers.

$\lg(n!) = \lg(1 \times 2 \times 3 \times \cdots \times (n-1)(n))$

$= \underbrace{\lg 1 + \lg 2 + \lg 3 + \cdots + \lg(n-1) + \lg(n)}_{n\text{項}} - ①$

$n \lg n = n \times \lg n = \underbrace{\lg n + \lg n + \cdots + \lg n}_{n\text{項}} - ②$

由①、②可看出：$\lg(n!) \le n \lg n$

$\Rightarrow \lg(n!) = O(n \lg n)$

$\lg(n!) = \lg(1) + \lg(2) + \cdots + \lg(\frac{n}{2}) + \cdots + \lg(n)$

$\lg(1) + \lg(2) + \cdots + \lg(n) \ge \lg(\frac{n}{2}) + \lg(\frac{n}{2}+1) + \cdots + \lg(n)$

$\ge \frac{n}{2} \lg(\frac{n}{2})$

$\Rightarrow \lg(n!) \ge \frac{n}{2} \lg(\frac{n}{2})$

$\Rightarrow \lg(n!) = \Omega(\frac{n}{2} \lg(\frac{n}{2}))$

$\Rightarrow \lg(n!) = \Omega(n \lg n)$

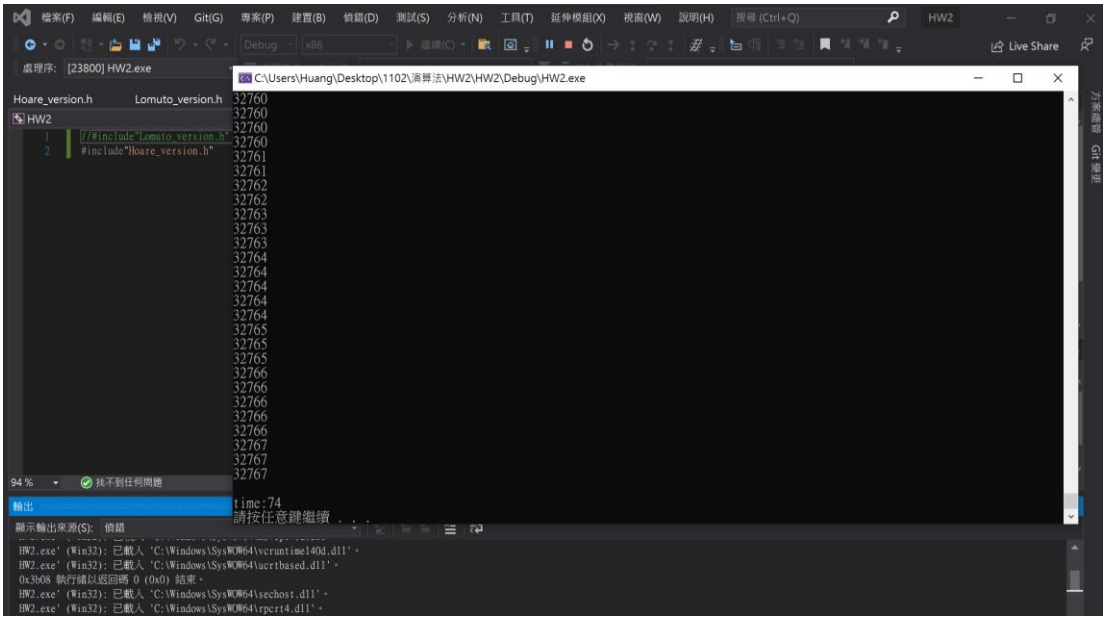
存在 $c_1, n_0 > 0$，當 $n > n_0$，$\lg(n!) \ge c_1 \cdot n \lg n$

存在 $c_2, n_0 > 0$，當 $n > n_0$，$\lg(n!) \le c_2 \cdot n \lg n$

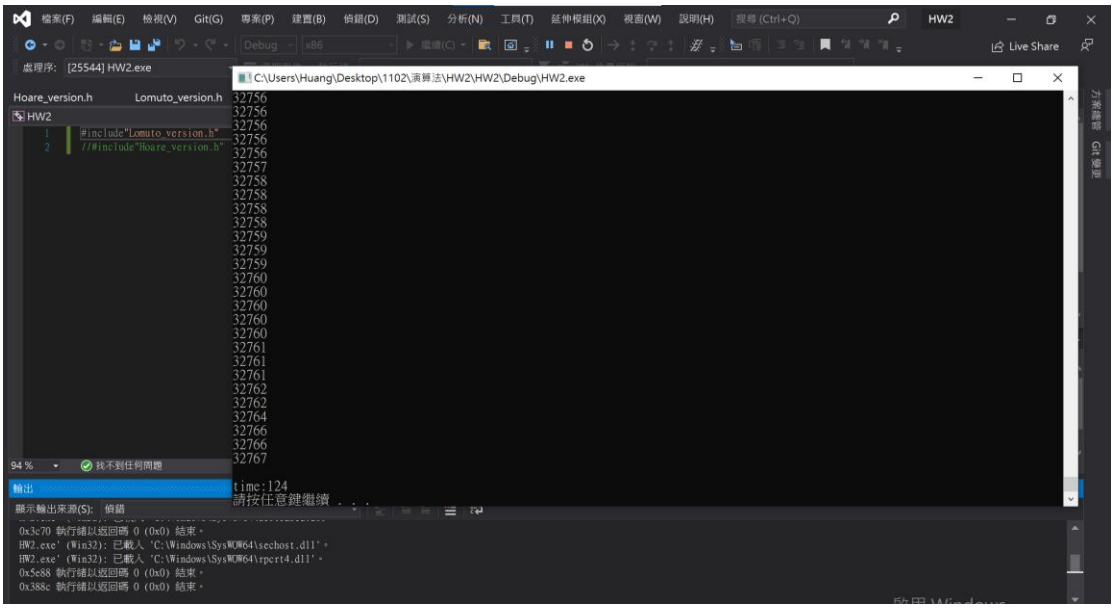
得 $c_1 \cdot n \lg n \le \lg(n!) \le c_2 \cdot n \lg n \Rightarrow \boxed{\lg(n!) = \Theta(n \lg n)}$

Problem 2.2. (bonus, up to 3 points) Compare the efficiency of different Quicksort implementations such as (1) the code you wrote for your data structure class (from Hoare in 1960, page 185 or shown below); or (2) the code taught by us (also known as Lomuto's version). You should have some trials and report their performance difference. Moreover, it is encouraged that you can provide some reasons why we may have such performance difference.

(1) Hoare's version : 74ms



(2) Lomuto's version : 124ms

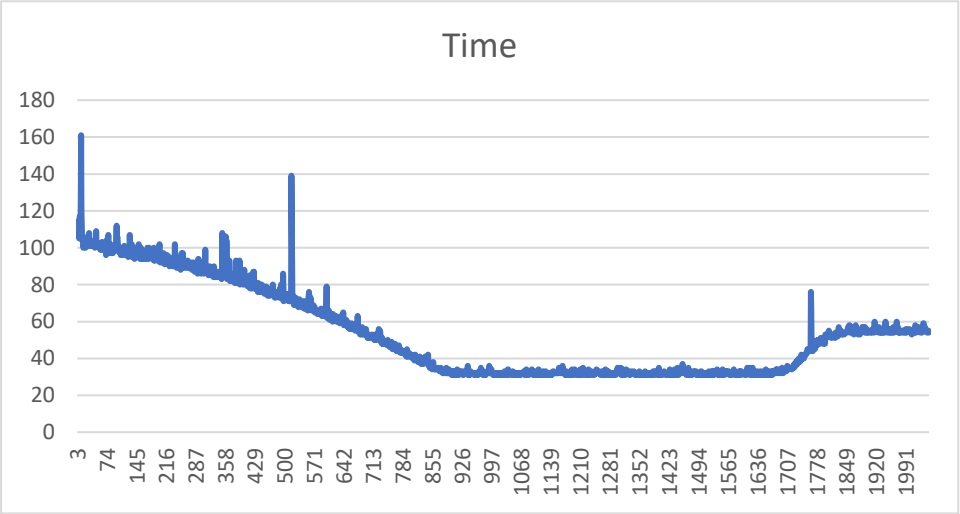


(2)比(1)慢很多，推測是(2)經過的判斷的 if 比(1)還多次。

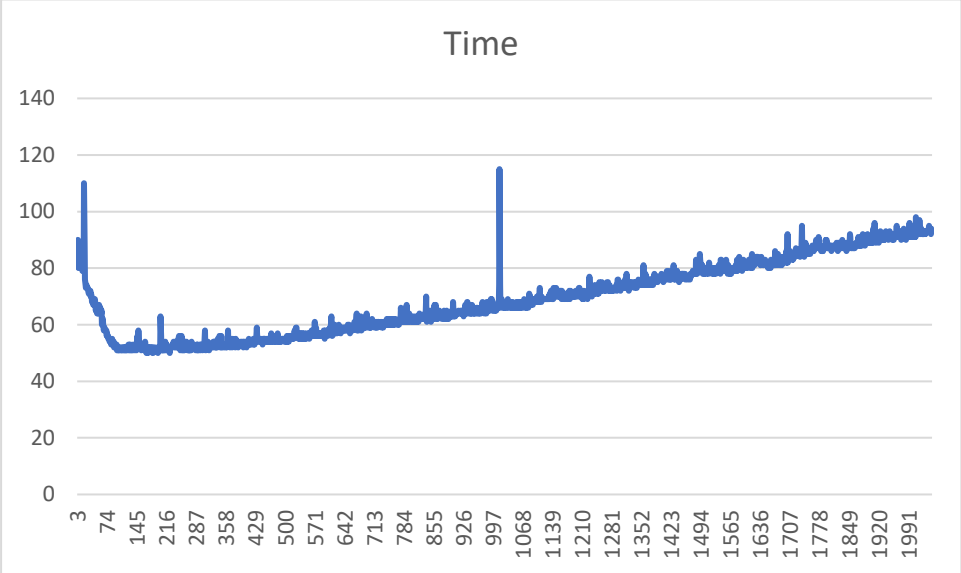
Problem 2.3. (bonus, up to 3 points) In slide No. 13, confirm the performance of the improved quicksort based on MEDIAN3 and small files sorted by insertion sort. You can give your best choice of M and suggest an amount of time reduction for the choice. A figure with the performance curve is helpful.

在參考 slide No. 13 後，我設定了一個 int original_num[87878] 放數字，並使用 srand(time(0))製造隨機 87878 個數字做 sorting，經過反覆測試後，發現數字的差異性對 M 也有影響，因此先測試不同組數字跑出來的結果:

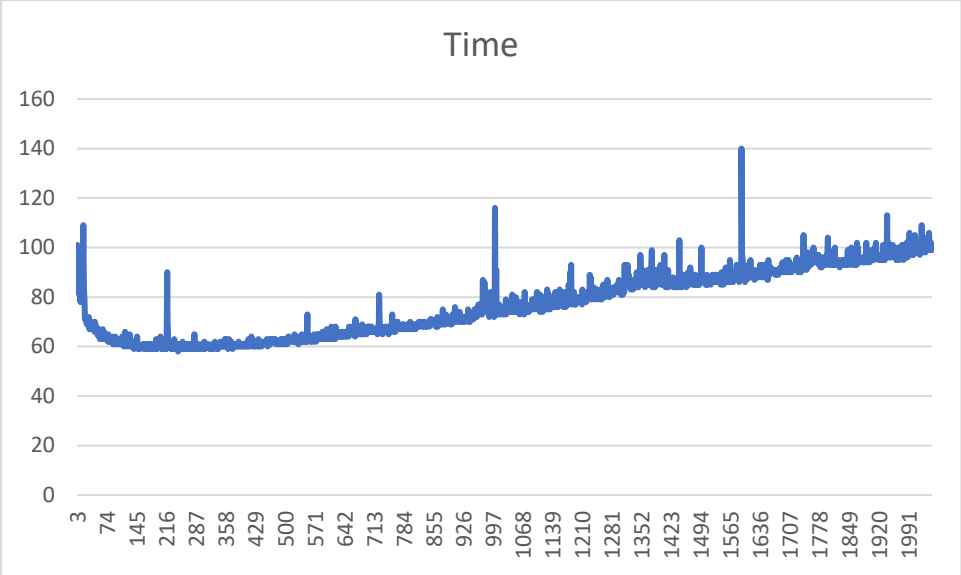
1. rand()%100 = 數字在 0~99 之間 (參考 TEST_99)



2. rand()%1000 = 數字在 0~999 之間 (參考 TEST_999)

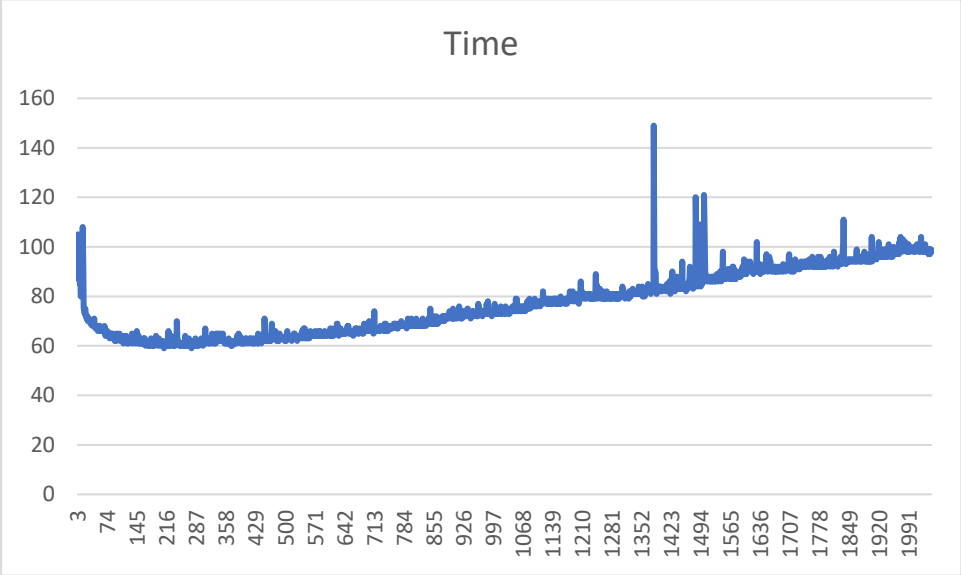


3. rand()%10000 = 數字在 0~9999 之間 (參考 TEST_9999)



經過上述測試發現除了在處理差異性最小的 1 之外，2 和 3 最好的表現大約都落在 74~145 之間。

接下來進行完全隨機數測試 (參考 TEST_RAND)



可以發現也是差不多的區間，多次測試後估計 M 值約 74~145。