Algorithm HW4_report

a. 程式說明:

這題為 Randomized quicksort，透過 random 找出區間內哪一個要是 pivot，並將它與最後一筆資料互換，之後再處理原本的 quicksort，如此一來，可以避免 pivot 永遠都是一串 data 的最後一筆資料，若它是已經排列好的(由小到大或由大到小)，會導致每次選到的 pivot 都是最大的或是最小的，如此增加時間複雜度。

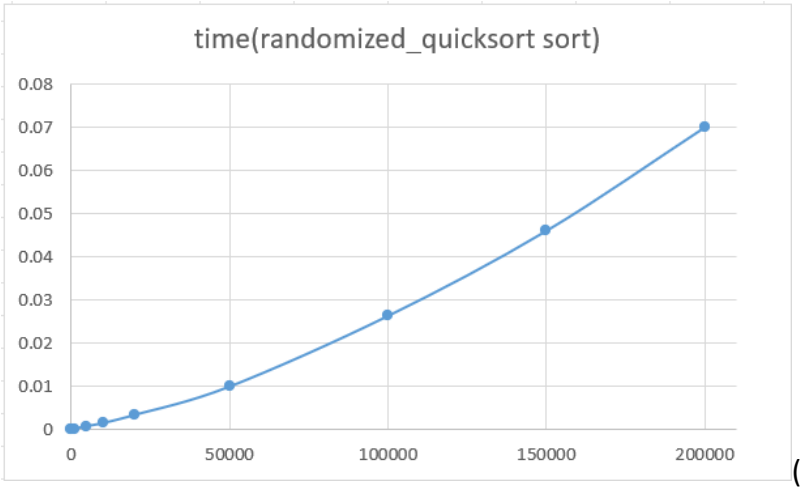
b. 程式結果:

(單位為秒)

size	50	100	1000	5000	10000	20000	50000	100000	150000	200000
time(insertion sort)	0.000013	0.000039	0.001663	0.040805	0.140099	0.524287	3.410914	13.35529	31.90713	63.51919
time(merge sort)	0.000041	0.000064	0.000309	0.002573	0.002573	0.005326	0.015128	0.03257	0.051484	0.077255
time(heap sort)	0.000013	0.00028	0.000209	0.002022	0.002022	0.004657	0.012633	0.030114	0.049423	0.06966
time(randomized_quicksort sort	0.000004	0.00001	0.000113	0.000753	0.001425	0.003307	0.009874	0.026236	0.04597	0.069975

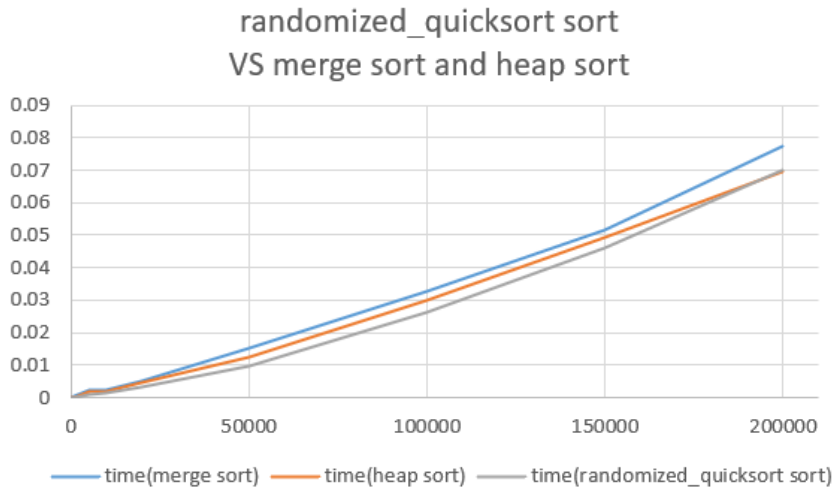
```
hong@LAPTOP-LERLMUM9:/mnt/c/Users/蔡東宏/Desktop/大二/Introduction to Algorithm/HW4$ ./a.out
Please cin data size(cin negative number to quit) : 50
randomized_quicksort sort sort during time: 0.000004s
Please cin data size(cin negative number to quit) : 100
randomized_quicksort sort sort during time: 0.000010s
Please cin data size(cin negative number to quit) : 1000
randomized_quicksort sort sort during time: 0.000113s
Please cin data size(cin negative number to quit) : 5000
randomized_quicksort sort sort during time: 0.000753s
Please cin data size(cin negative number to quit) : 10000
randomized_quicksort sort sort during time: 0.001425s
Please cin data size(cin negative number to quit) : 20000
randomized_quicksort sort sort during time: 0.003307s
Please cin data size(cin negative number to quit) : 50000
randomized_quicksort sort sort during time: 0.009874s
Please cin data size(cin negative number to quit) : 100000
randomized_quicksort sort sort during time: 0.026236s
Please cin data size(cin negative number to quit) : 150000
randomized_quicksort sort sort during time: 0.045970s
Please cin data size(cin negative number to quit) : 200000
randomized_quicksort sort sort during time: 0.069975s
```

Randomize quicksort:



(圖一)

與 heap sort 和 merge sort 以及 insertion sort 的比較:



(圖二)

c. 程式結果說明:

1. 觀察圖二，我們可以發現 randomized quicksort 執行時間與 merge sort 以及 heap sort 的執行時間與差不多，所以我們可以得知它的時間複雜度與 heap sort 以及 merge sort 一樣為 O(nlogn)。
2. Randomized quicksort 與一般的 quicksort 最大的差異是 pivot 的選取，Randomized quicksort 的選取方式是每次都在那串陣列隨機找一個當作 pivot 並將它交換到最後一個位置，而一般的 quicksort 是直接選最後一個當 pivot，用 Randomized quicksort 可以避免要排序的陣列是已經排序好的或是逆向排序好的，如此一來，選到的 pivot 就不會永遠是最大值也不會是最小值。
3. 雖然三者的排序執行是間都差不多，但仔細觀察可以發現 randomized quicksort 是三者裡面最快的，無論 size 多少。
4. 由 HW1 我們可以得知 insertion sort 的時間複雜度為 O(n^2)，它執行的速度遠不及時間複雜度為 O(nlogn)的 randomized_quicksort 及 heap sort 和 merge sort。
5. 這次取得時間也是使用 chrono(如同 HW3)，如此能夠更精確地抓到時間值，方便計算時間複雜度。