Algorithm HW3_report

a. 程式說明:

本次作業為做一個 heap sort，並將 heap sort 的執行結果與 HW1 的 insertion sort 和 merge sort 做比較，比較程式執行時間以及時間複雜度的差異。運用 oop 的概念，我建立了一個物件 heap，每一筆 data insert 的時侯，會根據它的值進行 reheap up，當所有資料都 insert 完成後，再將所有資料 reheap down 並存在某一個陣列裡，如此一來，heap sort 即完成，此陣列即為排好序的陣列。

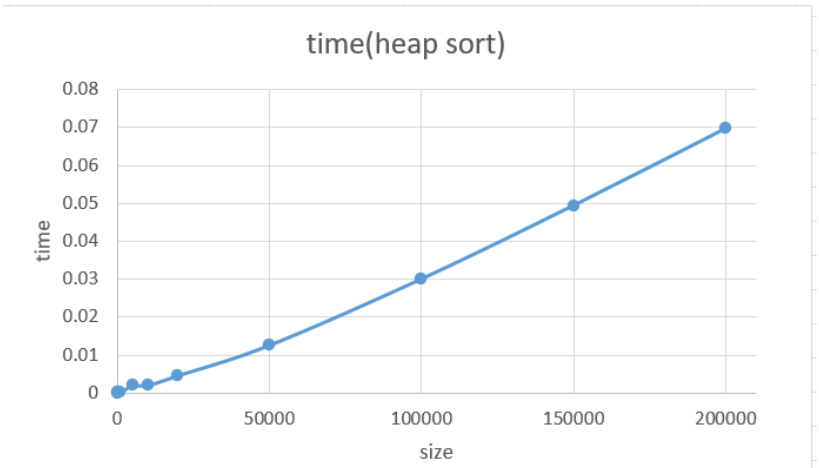
b. 程式結果:

程式執行時間單位(s):

size	50	100	1000	5000	10000	20000	50000	100000	150000	200000
time(insertion sort)	0.000013	0.000039	0.00166	0.04081	0.1401	0.52429	3.41091	13.3553	31.9071	63.5192
time(merge sort)	0.000041	0.000064	0.00031	0.00257	0.00257	0.00533	0.01513	0.03257	0.05148	0.07726
time(heap sort)	0.000013	0.00028	0.00021	0.00202	0.00202	0.00466	0.01263	0.03011	0.04942	0.06966

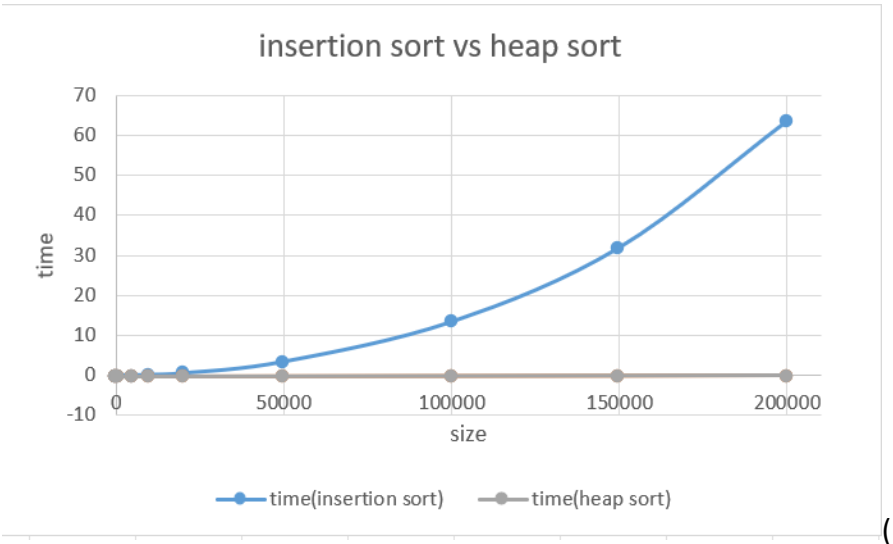
```
hong@LAPTOP-LERLMUM9:/mnt/c/Users/蔡東宏/Desktop/大二/Introduction to Algorithm/HW3$ ./a.out
Please cin data size(cin negative number to quit) : 50
insertion sort during time: 0.000013s
merge sort during time: 0.000041s
heap sort during time: 0.000013s
Please cin data size (cin negative number to quit) : 100
insertion sort during time: 0.000039s
merge sort during time: 0.000064s
heap sort during time: 0.000028s
Please cin data size (cin negative number to quit) : 1000
insertion sort during time: 0.001663s
merge sort during time: 0.000309s
heap sort during time: 0.000209s
Please cin data size (cin negative number to quit) : 5000
insertion sort during time: 0.040805s
merge sort during time: 0.001188s
heap sort during time: 0.000913s
Please cin data size (cin negative number to quit) : 10000
insertion sort during time: 0.140099s
merge sort during time: 0.002573s
heap sort during time: 0.002022s
Please cin data size (cin negative number to quit) : 20000
insertion sort during time: 0.524287s
merge sort during time: 0.005326s
heap sort during time: 0.004657s
Please cin data size (cin negative number to quit) : 50000
insertion sort during time: 3.410914s
merge sort during time: 0.015128s
heap sort during time: 0.012633s
Please cin data size (cin negative number to quit) : 100000
insertion sort during time: 13.355291s
merge sort during time: 0.032570s
heap sort during time: 0.030114s
Please cin data size (cin negative number to quit) : 150000
insertion sort during time: 31.907128s
merge sort during time: 0.051484s
heap sort during time: 0.049423s
```

Heap sort:



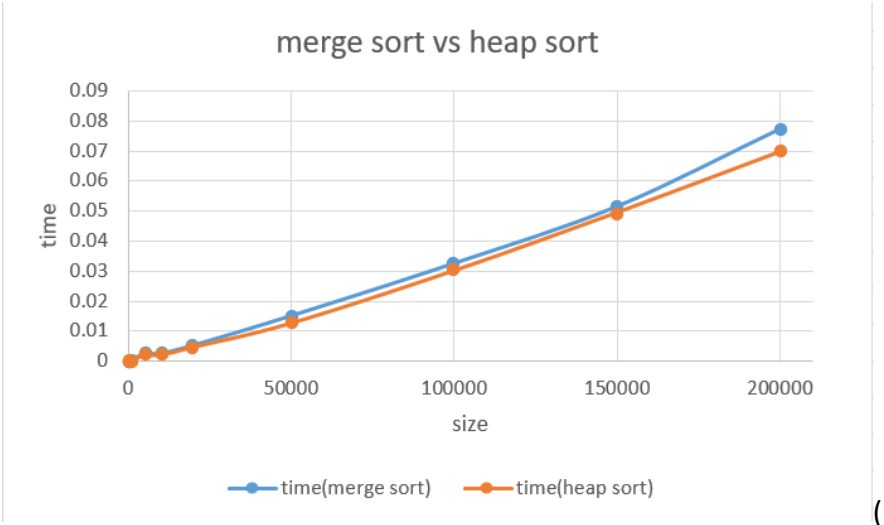
(圖一)

與 Insertion sort 比較:



(圖二)

與 merge sort 比較:



(圖三)

c. 程式結果說明:

1. 觀察圖三，我們可以發現 heap sort 的時間複雜度與 merge sort 的時間複雜度一樣，為 O(nlogn)，在相同 size 下，它們的執行時間幾乎一模一樣。
2.觀察圖二和圖三，insertion sort 在 size 很大的時候，程式執行時間非常長，遠大於其他兩種排序方法，所以我們可以得知，insertion sort 的排序方法非常沒有效率，不及 heap sort 以及 merge sort。
3.當 size 很小時，時間最慢的為 merge sort 而不是複雜度最高的 insertion sort，但隨著 size 慢慢變大時，insertion sort 的執行時間會超過 merge sort(大約 1000)。但無論如何，heap sort 的執行時間都低於 insertion sort 和 merge sort，是最有效率的排序方法。
4.heap sort 的 best case 為原本的資料已經排好序，此時的時間複雜度為 O(nlogn)，而 heap sort 的 worst case 為原本的資料反向排序，此時的時間複雜度也為 O(nlogn)。
5.在建立堆疊這個資料結構時，每一筆資料在 insert 的時候都需要 reheap up，若 reheap 到堆疊的最上面時，他需要做 logn 次，所以若有 n 筆資料 Insert 的時候，就需要做 nlogn，所以它的時間複雜度為 O(nlogn)。

d. 遇到困難:

一開始我使用 clock()函式去得到 sort 開始和結束的時間，並將它們相減得到 sort 需要的時間，但後來發現在 size 很小的時候，程式執行太快，且 Clock 抓時間不太準確，所以得到的 duration 都為 0，所以我後來上網查到了 chrono(用法如下)，他可以抓到更精確的時間，如此一來，將 sort 結束的時間減掉開始的時間，得到 sort 的時間非常精確，且不會等於 0，方便我們計算時間複雜度。

```
164         auto begin = chrono::high_resolution_clock::now();
165         insertion_sort(data,size);
166         //auto finish=clock();
167         auto finish = chrono::high_resolution_clock::now();
168         chrono::duration<double, ratio<1, 1>> dur = chrono::duration(finish - begin);
169         //cout << dur.count() << "s\n";
```

Reference: https://blog.csdn.net/cw_hello1/article/details/66476290