Homework 3

109610025 陳品妍

Comparison of insertion/merge/heap sort:

Size/time	100	1,000	10,000	100,000	200,000	300,000
Insertion sort	0s	0s	0.095s	3.083s	11.803s	29.558s
Merge sort	0s	0s	0.001s	0.016s	0.04s	0.061s
Heap sort	0s	0s	0s	0.015s	0.034s	0.05s

<table>
<tr><td>

```
please enter array size: 100
before sorting:

after insertion sort:

Time is measured by insertion sort: 0s
after merge sort:

Time is measured by merge sort: 0s
after heap sort:

Time is measured by heap sort: 0s
```

</td><td>

```
please enter array size: 1000
before sorting:

after insertion sort:

Time is measured by insertion sort: 0s
after merge sort:

Time is measured by merge sort: 0s
after heap sort:

Time is measured by heap sort: 0s
```

</td></tr>
<tr><td>100</td><td>1,000</td></tr>
<tr><td>

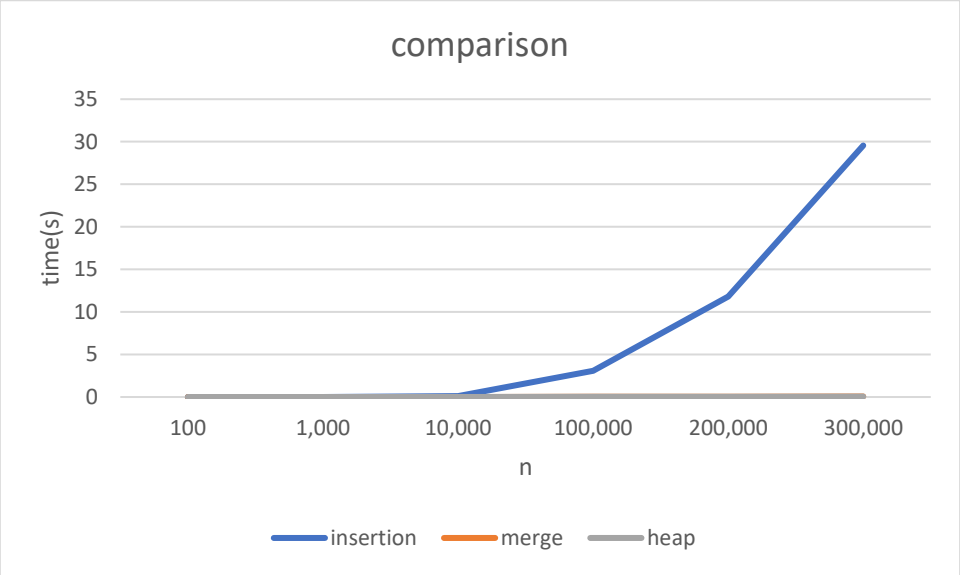
```
please enter array size: 10000
before sorting:

after insertion sort:

Time is measured by insertion sort: 0.095s
after merge sort:

Time is measured by merge sort: 0.001s
after heap sort:

Time is measured by merge sort: 0s
```

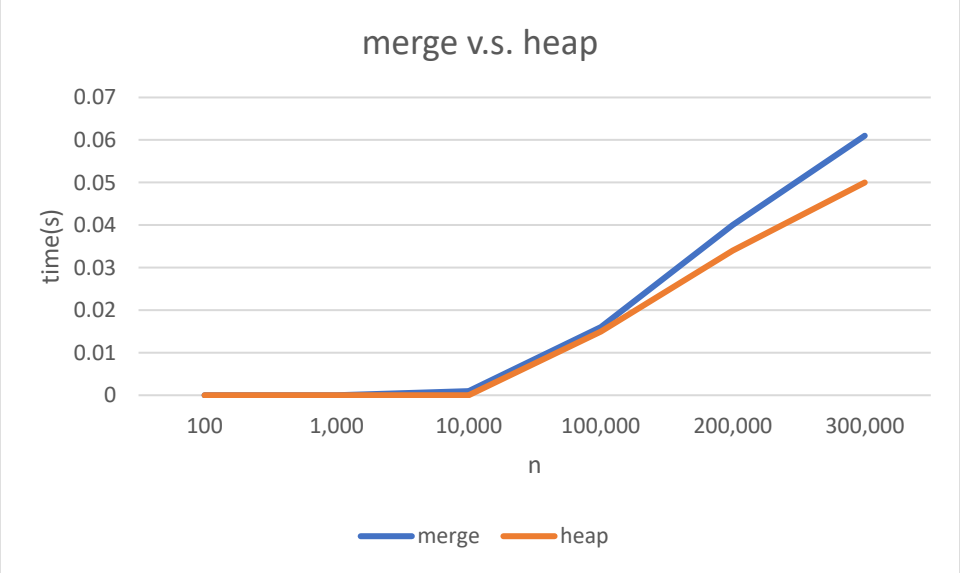
</td><td>

```
please enter array size: 100000
before sorting:

after insertion sort:

Time is measured by insertion sort: 3.083s
after merge sort:

Time is measured by merge sort: 0.016s
after heap sort:

Time is measured by heap sort: 0.015s
```

</td></tr>
<tr><td>10,000</td><td>100,000</td></tr>
</table>

please enter array size: 200000 before sorting: after insertion sort: Time is measured by insertion sort: 11.803s after merge sort: Time is measured by merge sort: 0.04s after heap sort: Time is measured by heap sort: 0.031s	please enter array size: 300000 before sorting: after insertion sort: Time is measured by insertion sort: 29.558s after merge sort: Time is measured by merge sort: 0.061s after heap sort: Time is measured by heap sort: 0.05s
200,000	300,000





100000 以及 200000:

$$\frac{0.034}{0.015} = 2.267 \rightarrow \frac{200000\log 200000}{100000\log 100000} = 2.12(O(nlogn))$$

100000 以及 300000:

$$\frac{0.05}{0.015} = 3.34 \rightarrow \frac{300000\log 300000}{100000\log 100000} = 3.286(O(nlogn))$$

100000 以及 400000:

$$\frac{0.095}{0.015} = 6.33 \rightarrow \frac{400000\log 400000}{100000\log 100000} = 4.4816(O(nlogn))$$

說明:

大致上來看 heap sort 符合課堂上所提到的時間複雜度 O(nlogn)。但可以發現隨著樣本變大，實際的執行時間越來越遠離上述的時間複雜度。可能的原因乃在不是全部的動作在 main 函式裡，而是一直遞迴呼叫下一個函式，需要記錄離開函式的資訊，跳回來時在查找上一次離開的地方，因此執行上會比理論稍微差一點。或者是也會跟進去的資料是有序還是倒敘亦或者是隨機分布的有差。時間複雜度 O(nlogn)是指平均下來的時間複雜度，因此一次實驗可能會有偏差。

實驗發現 heap sort 普遍比 merge sort 表現要突出，推測可能是因為 heap sort 遞迴冊數比 merge sort 還要少，在計算機組織的角度來說，減少跳出次數可以避免記憶體查找的時間。

程式說明:

首先執行時，系統會提式輸入需要的 array size 大小，如果需要看到輸出結果是否正確，可以將註解消掉。其他有關於演算法的架構大致上都是跟 pseudo code 相似，唯有在 index 修改以及創造一個 array 來放最大值。

程式正確性(以 10 為範例):

```
after heap sort:
31136   30783   26764   22790   22389   21710   19626   16952   4893    2550
```