## Homework 4

陳品妍 109610025

# Comparison of insertion/merge/heap sort/quicksort:

Size/time	100	1,000	10,000	100,000	200,000	300,000
Insertion sort	0s	0s	0.039s	2.63s	11.073s	36.037s
Merge sort	0s	0s	0.002s	0.02s	0.034s	0.09s
Heap sort	0s	0s	0s	0.02s	0.047s	0.062s
r-Quicksort	0s	0s	0s	0.017s	0.031s	0.035s

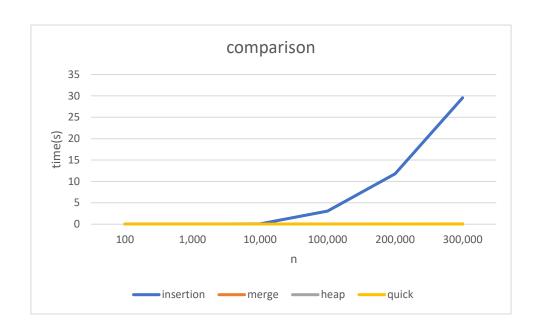
please enter array size: 100 before sorting:	please enter array size: 1000 before sorting:
after insertion sort:	after insertion sort:
Time is measured by insertion sort: 0s after merge sort:	Time is measured by insertion sort: 0s after merge sort:
Time is measured by merge sort: 0s after heap sort:	Time is measured by merge sort: 0s after heap sort:
Time is measured by heap sort: 0s after quick sort:	Time is measured by heap sort: 0s after quick sort:
Time is measured by quick sort: 0s	Time is measured by quick sort: 0s
100	1,000

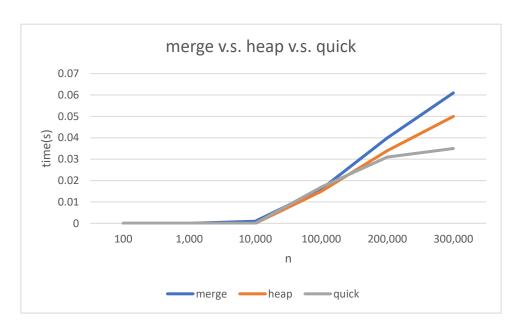
please enter array size: 100000 please enter array size: 10000 before sorting: before sorting: after insertion sort: after insertion sort: Time is measured by insertion sort: 2.63s Time is measured by insertion sort: 0.039s after merge sort: after merge sort: Time is measured by merge sort: 0.02s Time is measured by merge sort: 0.002s after heap sort: after heap sort: Time is measured by heap sort: 0.017s Time is measured by heap sort: 0s after quick sort: after quick sort: Time is measured by quick sort: 0.01s Time is measured by quick sort: 0s 10,000 100,000 please enter array size: 300000 please enter array size: 200000 before sorting: before sorting: after insertion sort: after insertion sort: Time is measured by insertion sort: 36.037s Time is measured by insertion sort: 11.073s after merge sort: after merge sort: Time is measured by merge sort: 0.09s Time is measured by merge sort: 0.034s after heap sort: after heap sort: Time is measured by heap sort: 0.062s Time is measured by heap sort: 0.047s after quick sort: after quick sort: Time is measured by quick sort: 0.035s

300,000

Time is measured by quick sort: 0.031s

200,000





100000 以及 200000:

$$\frac{0.031}{0.017} = 1.823 \rightarrow \frac{200000}{100000} = 2.12(O(nlogn))$$

100000 以及 300000:

$$\frac{0.035}{0.017} = 2.058 \rightarrow \frac{300000}{100000} = 3.286(O(n))$$

說明:

首先,因為數字是在特定範圍亂數產生,某種程度上可以視作已經經過 randomized partition,因此沒有特定再去對 partition 做 randomized。然而,實驗之後發現 quicksort 的表現比較不穩定,沒有達到課堂所述的 O(nlogn)。因為 quicksort 的表現會因為輸入資料的亂度影響,即亂度越高,執行效果越佳。所

以當我們在討論他的時間複雜度時,其表現可能因為亂度不一,而導致其資料之間的差異這麼的大。因此後來才又針對 partition 再去用亂數處理。透過 randomized partition,的確發現 randomized quicksort 的表現趨於穩定,表現都比 heap sort 以及 merge sort 好。在所有的執行時間當中,可以發現 insertion sort 是最差的,其次 merge sort 以及 heap sort。考慮到其時間複雜度以及條件限制,我們可以合理推論,insertion sort 適合用在資料較少的排序中,以避免資料太大使執行時間過大的狀況出現。Merge sort 則是可以很穩定的表現,隨著資料上升。Heap sort 則是因為不會用到額外空間(即原地排列),適合對內部記憶體有限制的情況使用。而 quicksort 則要保證其輸入資料的亂度,否則會出現最壞的狀況,即 $O(n^2)$ 。

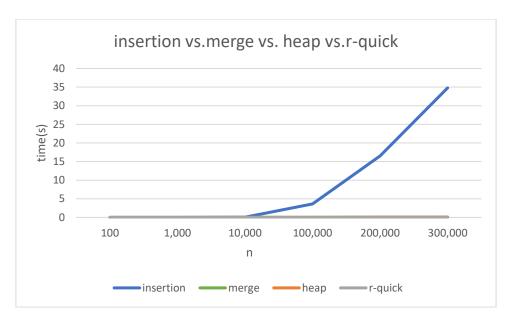
Algorithms	Advantages	Disadvantages		
Insertion	· 針對小數據或部分有序	· 對於大型未知數據性		
	數列,表現較佳	能差		
	・ 實現簡單			
Merge	• 穩定的時間複雜度,保	・需要額外記憶體空間		
	持一致性能(不受資料原	來儲存中間結果		
	本排序影響)	· 需要遞迴跟合併的操		
		作,實現較為複雜		
Неар	· 原地排序,不須額外用	・ 演算法牽涉交會元素		
	到其他記憶體	的位置,可能導致實		
	· 性能穩定(最差情況仍然	際效率不如其他演算		
	為 nlogn)	法。		
Randomized quick	・ 實現簡單	・需要保證輸入資料的		
	· 具有 nlogn 的時間複雜	亂度,否則時間複雜		
	度	度變成 n^2		

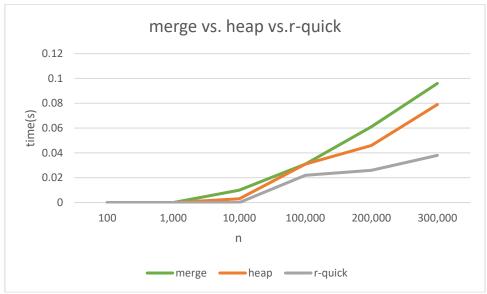
#### 加上 randomized partition 之執行結果

#### Comparison between insertion sort/merge sort/heap sort/randomized sort:

			<i>.</i>	· · ·		
Size/time	100	1,000	10,000	100,000	200,000	300,000
Insertion sort	0s	0s	0.058s	3.608s	16.511s	34.793s
Merge sort	0s	0s	0.01s	0.031s	0.061s	0.096s
Heap sort	0s	0s	0.003s	0.031s	0.046s	0.079s
r-Quicksort	0s	0s	0s	0.022s	0.026s	0.038s

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 nandomized quick sort:  Time is measured by heap sort: 0s after randomized quick sort:  Time is measured by insertion sort: 0s after nandomized quick sort:  Time is measured by heap sort: 0s after randomized quick sort:  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort: 0s  Time is measured by nandomized quick sort: 0s  Time is measured by insertion sort:  Time is measured by insertion sort:  Time is measured by insertion sort:  Time is measured by merge sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort:  Time is measured by merge sort: 0.022s  Time is measured by merge sort: 0.051s after insertion sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by merge sort: 0.005s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured by merge sort: 0.006s after merge sort:  Time is measured		<del>_</del>	
Time is measured by insertion sort: 0s after merge sort:  Time is measured by merge sort: 0s after merge sort:  Time is measured by merge sort: 0s after heap sort:  Time is measured by heap sort: 0s after heap sort:  Time is measured by heap sort: 0s after randomized quick sort:  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort: 0s  Time is measured by insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort: 0.022s  Time is measured by randomized quick sort: 0.025s  Time is measured by randomized quick sort: 0.061s after merge sort:  Time is measured by insertion sort: 0.061s after merge sort:  Time is measured by heap sort: 0.066s after heap sort:  Time is measured by heap sort: 0.069s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s			
after merge sort:  Time is measured by merge sort: 0s after heap sort:  Time is measured by heap sort: 0s after heap sort:  Time is measured by heap sort: 0s after heap sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0s  Time is measured by insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort: 0s  Time is measured by randomized quick sort: 0s  Time is measured by insertion sort: 16.511s after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by heap sort: 0.061s after heap sort:  Time is measured by heap sort: 0.061s after heap sort:  Time is measured by heap sort: 0.066s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.003s	after insertion sort:	after insertion sort:	
after heap sort:  Time is measured by heap sort: 0s after randomized quick sort:  Time is measured by randomized quick sort: 0s  100  1,000  please enter array size: 10000 before sorting:  after insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by heap sort: 0.02ss  Time is measured by heap sort: 0.02ss  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0.02cs  Time is measured by randomized quick sort: 0.02ss  Time is measured by randomized quick sort: 0.02ss  Time is measured by randomized quick sort: 0.0000  please enter array size: 200000 before sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort: 0.096s after randomized quick sort: 0.093s  Time is measured by randomized quick sort: 0.03ss			
after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort:  100  1,000  please enter array size: 10000 before sorting:  after insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by insertion sort:  Time is measured by insertion sort: 16.511s after heap sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by merge sort: 0.046s after heap sort:  Time is measured by heap sort: 0.046s after nandomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized q	, ,		
Inhe is measured by randomized quick sort: 0s  100  please enter array size: 10000 before sorting:  after insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort: 0s  10,000  please enter array size: 200000 before sorting:  after insertion sort:  Time is measured by randomized quick sort: 0s  10,000  please enter array size: 200000 before sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s			
please enter array size: 10000 before sorting:  after insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort: 0.002s  Time is measured by randomized quick sort: 0.002s  Time is measured by insertion sort: 16.511s after insertion sort:  Time is measured by insertion sort: 34.793s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s  Time is measured by randomized quick sort: 0.038s	Time is measured by randomized quick sort: 0s	Time is measured by randomized quick sort: 0s	
before sorting:  after insertion sort:  Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.031s after randomized quick sort:  Time is measured by heap sort: 0.031s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0.022s  Time is measured by randomized quick sort: 0.022s  Time is measured by insertion sort:  Time is measured by insertion sort:  Time is measured by insertion sort: 300000 before sorting:  after insertion sort:  Time is measured by insertion sort: 34.793s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s	100	1,000	
Time is measured by insertion sort: 0.058s after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.031s after randomized quick sort:  Time is measured by randomized quick sort: 0.022s  Time is measured by randomized quick sort: 0.022s  Time is measured by randomized quick sort: 0.022s  Time is measured by insertion sort:  Time is measured by insertion sort: 16.511s after insertion sort:  Time is measured by insertion sort: 34.793s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s			
after merge sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized duick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0.022s  Time is measured by randomized quick sort: 0.022s  Time is measured by insertion sort:  Time is measured by insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by merge sort: 0.066s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.026s  Time is measured by randomized quick sort: 0.038s	after insertion sort:	after insertion sort:	
Time is measured by merge sort: 0.01s after heap sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by heap sort: 0.003s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0.002s  Time is measured by randomized quick sort: 0.002s  Time is measured by randomized quick sort: 0.002s  Time is measured by insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s	-	*	
Time is measured by heap sort: 0.003s after randomized quick sort: Time is measured by randomized quick sort: Time is measured by randomized quick sort: Time is measured by randomized quick sort: 0.022s  Time is measured by randomized quick sort: 0s  10,000  please enter array size: 200000 before sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.026s  Time is measured by randomized quick sort: 0.038s	2 0	, ,	
Time is measured by randomized quick sort: 0s  10,000  please enter array size: 200000 before sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.026s  Time is measured by heap sort: 0.038s	·	· ·	
please enter array size: 200000 please enter array size: 200000 please enter array size: 300000 pefore sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.026s  Time is measured by randomized quick sort: 0.038s	·	Time is measured by randomized quick sort: 0.022s	
please enter array size: 200000 before sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by randomized quick sort: 0.026s  please enter array size: 300000 before sorting:  Time is measured by insertion sort: 34.793s after is measured by insertion sort: 34.793s after merge sort:  Time is measured by merge sort: 0.096s after heap sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s	Time is measured by randomized duick sort: 0s		
before sorting:  after insertion sort:  Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort: 0.038s	10,000	100,000	
Time is measured by insertion sort: 16.511s after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by merge sort: 0.096s after heap sort:  Time is measured by merge sort: 0.096s after heap sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by heap sort: 0.038s			
after merge sort:  Time is measured by merge sort: 0.061s after heap sort:  Time is measured by merge sort: 0.096s after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0.038s	after insertion sort:	after insertion sort:	
after heap sort:  Time is measured by heap sort: 0.046s after randomized quick sort:  Time is measured by heap sort: 0.079s after randomized quick sort:  Time is measured by randomized quick sort:  Time is measured by randomized quick sort: 0.038s			
after randomized quick sort:  Time is measured by randomized quick sort: 0.026s  after randomized quick sort:  Time is measured by randomized quick sort: 0.038s			
Time is measured by randomized quick sort; 0.026s		·	
200,000 300,000	Time is measured by randomized quick sort: 0.026s	Time is measured by randomized quick sort: 0.038s	
	200,000	300,000	





100000 以及 200000:

$$\frac{0.026}{0.022} = 1.18 \rightarrow \frac{200000}{100000} = 2.12(O(nlogn))$$

100000 以及 300000:

$$\frac{0.038}{0.022} = 1.72 \rightarrow \frac{300000}{100000} = 3.286(O(nlogn))$$

#### 說明:

加上 randomized 的確發現比沒有加上 randomized 的趨勢更加穩定,但不是很明顯。有可能是原數列就是亂數產生的,所以本質上已經是亂的。加上 randomized 是避免輸入資料有排序規則的情況。上方的所有 quicksort 執行結果都比原本 nlogn 小,猜測可能是因為剛好輸入的資料非常亂,所以讓 quicksort 的效率提升。因此使 deterministic 的結果比統計行為(nlogn)佳。

### Correctness of randomized quicksort:

以 n=10 為例

after randomized quick sort: 6221 14630 17514 17808 20661 21865 26469 26556 26715 31093