B10815057 Algorithms homework1

* 1. 不是max-heap
  2. 可以
  3. 因為只有資料6不符，與資料7交換(一次MAX-HEAPIFY)後就符合
  4. i > heapsize[A]/2，因為leaf node的數量是heapsize[A]/2取上高斯，所以i節點必定是leaf node，而leaf node自身已是max-heap，只需花費常數時間(判斷是否需要交換)就可完成MAX-HEAPIFY，因此沒有影響。
  5. 任意刪除一個heap節點，該節點不一定比最後節點還大，若是用大於刪除節點的最後節點替換，則會造成increase key，此時就必須使用HEAP-INCREASE-KEY來解決，若是一般狀況使用MAX-HEAPIFY即可解決，由於HEAP-INCREASE-KEY與MAX-HEAPIFY，時間複雜度皆為O(lgn)，故此演算法為O(lgn) 
  6. 最佳是insertion sort，因為insertion sort是將某一個資料由後向前掃描，放入正確位置，所以就算有新資料加入也不需要整個sorting重做，但selection與quick sort若有新資料加入，若不處理或重做，會導致最終結果錯誤，因此insertion sort相對其他2個演算法不需要額外處理的時間，時間複雜度會保持相同。
  7. 這三個演算法都能找到最大的三筆資料
  8. 因為這三個演算法都能完整將sequence排序，然後取前三個元素就是最大的三筆資料

Insertion:O(n^2)從第二個元素到最後，每個元素會從後面往前掃過已排序的部分，一邊把資料往後搬，直到遇到比自己大的元素，就放在該元素之後，每個元素都插入過後，整個序列就是排序好的。



Selection: O(n^2)一開始的處理範圍為整個序列，每次將處理範圍中最大的資料移到處理範圍的最前面，然後將處理範圍縮減一格，直到處理範圍為0，因為每次都是找最大的，所以整個序列就會變成:第1大, 第2大, 第3大…第n大，這樣就是排序完成



Bubble: O(n^2)一開始的處理範圍為整個序列，每次比較相鄰的元素，會將越大的元素往前推，然後縮減處理範圍直到為0，與Selection sort類似，最後就排序完成



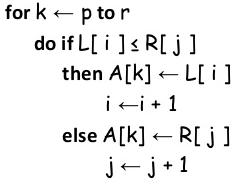
6.1

Insertion:此演算法是插入到第一個大於等於元素的後一位置，因此相同大小的元素不會被改變順序



Merge:會將序列遞迴得區分成許多對數字，然後在合併時完成排序，雖然序列被區分為好幾個部分，但部分之間彼此的順序還是保留著，因此遇到相同元素時，只需依照部分之間的順序即可。

如下圖，若L[i]==R[j]，則進入then處，取代A[k]的就是L[i]，而下次迴圈就會進入else處，所以L[i]在左，R[j]在右，仍然保持正確順序



Selection:每次迴圈會將範圍內最大的值放到前面，但由於可能是透過非相鄰的元素交換，所以相同大小的元素在這個過程中，就可能被破壞順序，舉例:序列5 8 5 2 9要由小排到大，第一遍選擇第1個元素5會和2交換，此時的2個5順序就被調換了，因此selection sort是不穩定的



Bubble:氣泡排序法的每次比較都是相鄰的元素在比，因此如果相同就不交換，即可不破壞順序



Heapsort:因為創建heap的過程與第一次拿出根結點並將最後節點放到根結點後，順序就會被破壞了，舉例:

原始:{1, 5, 2a, 3, 2b, 6, 2c}

創建heap後(array representation): {1, 2b, 2a, 3, 5, 6, 2c}

排序後結果: {1, 2c, 2b, 2a, 3, 5, 6}

Quicksort:每次將某個元素放到最終正確的位置，但在過程中，可能會使用到多次交換，因此順序非常容易被破壞，如下圖黑色方框處

