Prob. 8-3

05	0000000005	5	5#########
02	0000000002	2	2#########
0121	0000000121	121	121#######
015524	0000015524	15524	15524#####
011623	0000011623	11623	11623#####
099876126	0099876126	99876126	99876126##
066532123	0066532123	66532123	66532123##
023345129	0023345129	23345129	23345129##
1122222123	1122222123	1122222123	1122222123
(a)	(b)	(c)	(d)

a.

方法

1. 算出每個 integer 所擁有的 digit 數並依照這數量 sort
2. 紀錄 digits 數量改變的位置，以上圖(a)為例就是那些藍線的位置
3. 套用課本的 Radix Sort，從最後一位排回來，不過做以下修改
   A) 除了最長數列以外，每個數列前方都假想有個 0 存在
   B) Sort 依然是從最右邊的 digit 往左邊做，但並不是每次都做全部的數字，會慢慢減少，實際做的範圍請參考上圖(a)的淺色區域
4. 最後排完的結果就是答案

正確性

參考圖(b)，在第一次依照長度排完之後，若我們把前面補滿 0，再套用課本原始的 Radix Sort 其正確性是無須懷疑的，因此今天我們要證明的是，是否只要看到第一個補上的 0，之後不用再理他答案也會正確。

原始 Radix Sort 的過程，當看到 0 的時候，該輪排序一定會把 0 的那些數字都排到最上方，而當兩個以上數字都是 0 的時候，會依照原本兩個數字的順序排列(Stable Sort)。

稍微觀察我們不難發現，當一個數字做 Radix Sort 過程中，第一次看到補上的 0 後，那輪結束的位置，就會是這數字最後的位置。因為這數列上方的數字開頭一定也都是補上的 0，假若不是補上的 0，表示上面這個數字一開始就比我長，但是比我長這輪又和我同時是 0，應該會在我的下面而形成矛盾。(因為同時是 0 的話，會依照起始(依照長度排)的順序)

既然上方的數列全都是補上的 0，那日後不管怎麼排，這些人一定動都不會動，因爲會一直看到 0，既然如此那這些人根本就沒必要去理他，所以看到第一個補上的 0 之後，就可以不用再管這個數字，因爲他的位置已經對了，排剩下位置還沒確定的就行，故我們改過的 Radix Sort 正確性上是沒有問題的。

時間分析

1. 就算我一個一個 digit 看來算出長度，時間也不會超過 $O(n)$，之後排序由於長度是整數且不會超過 n，套用課本 Counting Sort 可以在 $O(n)$完成
2. 這些藍線的位置只要從上往下看一遍就可以得知，時間 $O(n)$
3. Radix Sort 每 sort 一列 digit 的時間是 $O(k + \sum)$，其中 k 是數字個數，$\sum$ 是數字範圍，上圖(a)可以看出全部做的數字個數是淺色區域的範圍，而且做的列數不會超過 n，所以總時間是 $O(n + n\sum + n) = O(n)$，其中我們補的假想 0 不會超過 n 個，$\sum$這邊是個常數。且定位的數字都必會位於上方連續，需要繼續算的數字都位於下方連續，根據一開始算的藍線位置，要 Implement 改過的 Radix Sort 是很容易的一件事情。
4. $O(1)$，實際上什麼事情也不用做

因此總時間 $O(n)$

b.

方法

1. 算出每個字串的長度數並依照長度 sort
2. 紀錄長度改變的位置，以上圖(c)爲例就是那些藍線的位置
3. 套用課本的 Radix Sort，從最後一位排回來，不過需做些修改，和 a 小題類似，但這次不補 0，全部數字往左靠，從最右側往左做，作的範圍請參考上圖(c)
4. 最後排完的結果就是答案

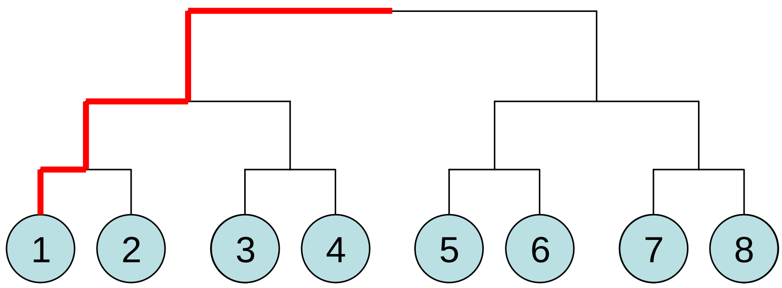
正確性

參考圖(d)，和 a 小題類似，在還沒看到#以外的字之前，就算去排他位置也不會動，不如不要做，其中#請視爲比任何字元都還要小的字元。

時間分析

和 a 小題幾乎一樣，所以省略，一樣是 $O(n)$

Ex. 9.1-1



為了解釋方便先假設$n = 2^m$，$\log n = m$

我們採取如上圖淘汰賽的方式比較，每比較一次就少一個人，總共有 n 個人，所以比較 $n - 1$ 次我們就可以很順利的找到最小的那個人。

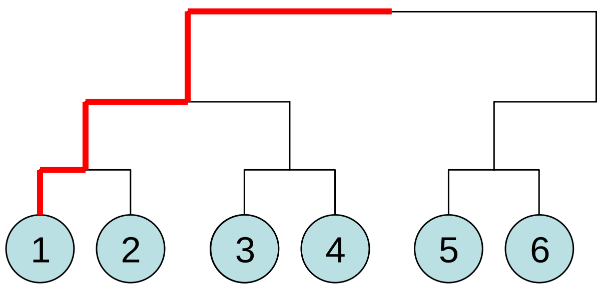
但是現在問題是如何找第二小的人，我們不難發現，第二小的人除非遇到最小的，才會被它幹掉而淘汰，因此有可能是第二小的，只有被最小幹掉的那些人有可能。

以上圖爲例，採用這種比較方式，最小的那個人(在這圖假設是一)，在成爲冠軍之前，只會幹掉 $\log n$ 個人，而這 $\log n$ 個人中藏有我們想要的第二小。

第二小的傢伙就是這 $\log n$ 個人中最小的那傢伙，這邊我們就隨便拿兩個比，小的留下，大的拋棄，每比一次可以拋棄一個人，因此比 $\log n - 1$ 次就會剩下最小的那個，也就是我們要找的第二小。

於是從頭到尾我們比較的次數是 $n - 1 + \log n - 1 = n + \log n - 2$

接下來要解決n並不剛好是 2^m的情況，這種情況我們就先畫一個最小但是大於n的 2^k比較樹，再把不存在的那些人拔掉，以下圖爲例，今天只有六個人的話，就先畫棵大小是八的，再把七八拔掉。



很明顯的我們可以看到高度不會超過$\lceil \log n \rceil$，所以 Worst Case 最小的數字頂多就幹掉了$\lceil \log n \rceil$這麼多人，因此和上述類似的分析我們可以得到只要比較$n + \lceil \log n \rceil - 2$次就可以找出第二小。

The second smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case!