The homework assignment is available at:
`http://www.jennylam.cc/courses/146-s17/homework13.html`

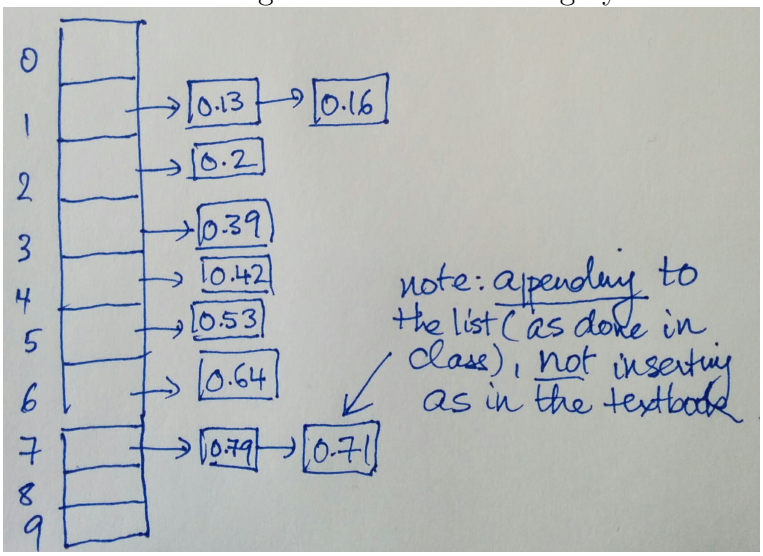
(a) (8.3-1) Using figure 8.3 of the textbook as a model, illustrate the operation of radix sort on the following list of English words: cow, dog, sea, rug, row, mob, box, tab, bar, ear, tar, dig, big, tea, now, fox.

Solution.

bucket sort by 3rd letter	by 2nd	by 1st
sea	tab	bar
tea	bar	big
mob	ear	box
tab	tar	cow
dog	sea	dig
rug	tea	dog
dig	dig	ear
big	big	fox
bar	mob	mob
bar	dog	now
ear	cow	row
tar	row	rug
row	now	sea
now	box	tab
box	fox	tar
fox	rug	tea

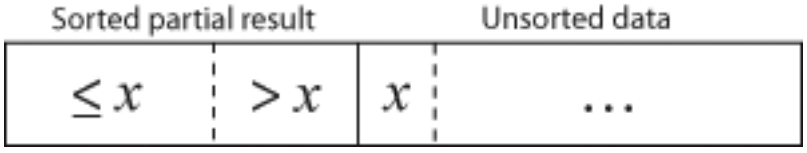
(b) Using figure 8.4 of the textbook as a model, illustrate the operation of bucket sort on the array $A = [0.79, 0.13, 0.16, 0.64, 0.39, 0.20, 0.89, 0.53, 0.71, 0.42]$

Solution. Assuming we are bucket-sorting by the first decimal digit:



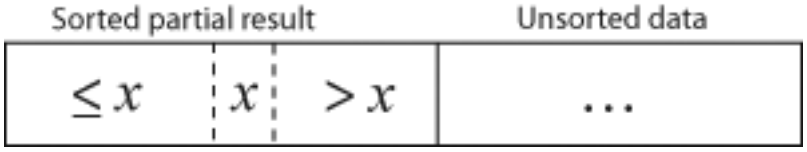
(c) Which of the following algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? Give a simple scheme that makes any comparison sort stable. How much additional time and space does your scheme entail?

Solution. Insertion sort is stable. To see this, recall that the invariant is:



(Image source: wikipedia)

and as a result of the "insertion step" of x:



(Image source: wikipedia)

Before this insertion, "x" is to the right of any equal value in the sorted partial result. After the insertion, "x" remains to the right of the sorted partial result that is $\leq x$. Inducting over all insertion steps, insertion sort is stable.

Mergesort is also stable. This relies on the merge step being stable (since this is the only step in which data moves around). This is true if the merge step is carefully implemented. First, the two input arrays must be listed in the right order: left array, followed by right array. Second, when two values are compared and equal, merge should select the one from the left array.

Heapsort is not stable. For example, consider the array [3,1,2,2]. The first step is to heapify this array, starting with a sift-down of 1 (draw the tree representation to see why this is true), which has the effect of swapping 1 and the last 2 in the array, to get [3,2,1,2]. So the two 2's reversed their relative order.

Quicksort is not stable. For example, the array [2,2,1]. During partitioning (at the top level of the recursion, the pivot (the first 2) will be swapped with 1. Therefore, the relative order between the first and second 2 is not preserved.

To make any sorting algorithm stable, add a second input to each key that represents the original position in the array, For example $[2, 2, 1]$ becomes $[(2, 0), (2, 1), (1, 2)]$. Then sort by the first (the original key) and use the second only as a tie breaker. This way $(2, 0) < (2, 1)$ so the first 2 will be to the left of the second 2, even if the original sorting algorithm is not necessarily stable. •

1. Define the "first drop problem", on an input array A of n numbers, to be the problem of finding the first position at which one number drops to a smaller number after it. More formally, the output should be the smallest index i for which $A[i] > A[i+1]$, or $i = n-1$ if no such index exists. For instance, on the input [5,9,2,4,1] the output should be 1, because

the first drop is from the values 9 to 2, and the value 9 occurs at position $A[1]$. But on the input [1,2,4,5,9] there is no first drop and the output should be 4. You may assume that no two input numbers are equal.

(a) When $n = 4$, how many different outcomes does the first drop problem have?

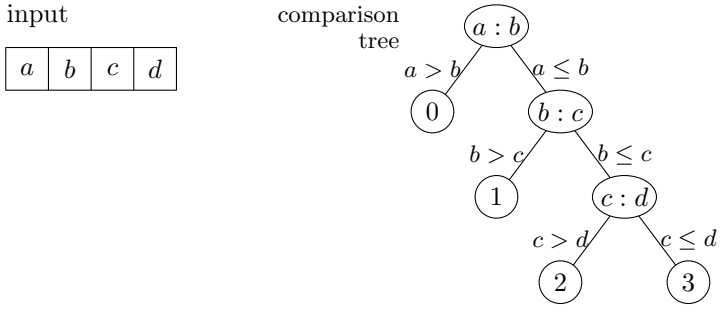
Solution. There are 4 outcomes, one for each index of the array. •

(b) Based only on your answer to part (a), and the $\lceil \log_2(\#\text{outcomes}) \rceil$ lower bound on the height of any binary comparison tree, how many comparisons would be necessary to correctly solve the first drop problem for $n = 4$?

Solution. A lower bound is $\lceil \log_2(\#\text{outcomes}) \rceil = \log_2 4 = 2$. •

(c) Draw a comparison tree that uses as few comparisons as possible (in the worst case) to solve the first drop problem for $n = 4$. How many comparisons in the worst case does your tree use?

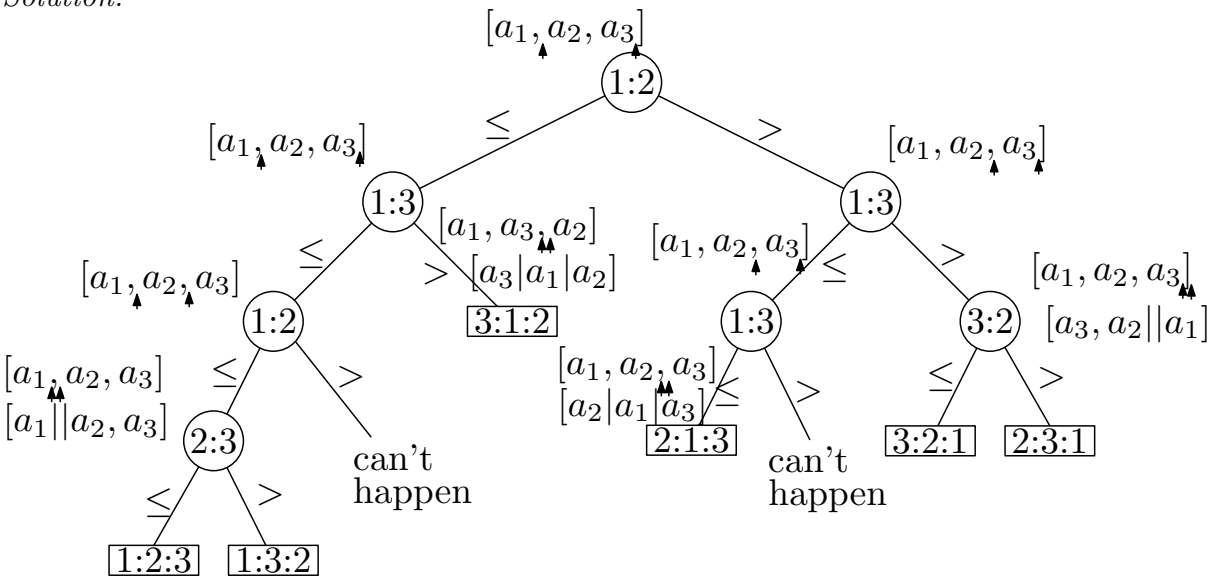
Solution. The following tree does 3 comparisons in the worst case.



•

2. Draw a decision/comparison tree for quicksort on inputs of length 3.

Solution.



•

3. Find a weighted undirected graph for which the MST-doubling heuristic can generate a tour that is longer than optimal by a factor of at least 5/3. Show both the optimal tour and the MST-doubling tour for your example. (You may use the simplest version of MST-doubling, without shortcutting repeated vertices.)

   Solution. In a cycle of length 1000 in which all edges have a weight of 1, the MST-doubling heuristic gives a tour with length 1998 (twice the length of a path through all the vertices), whereas the optimal solution has length 1000 (the length of the cycle). The approximation ratio is therefore 1998/1000 which is greater than 5/3. •