Solutions Topic 10b, 10/02: Linear and Stable Sorts

1. Stable Sort

(a) How can you modify any sorting algorithm to make it stable? Be specific about the additional data structures and changes to the algorithms needed.

> **Solution:** We need to record information about the original ordering, but we do not want to modify the original data. (Tricks like shifting the bits over and using lower order bits to record position will also work, but assume particular key data types.) A simple approach is to make the keys into (key, position) tuples and providing a comparison operator for these objects. The comparison operator will compare first on key, and then if they are equal compare on position to break the tie. This is modular, not requiring changing the sort code. A similar approach is to set up a parallel array of positions I[i] = i, and manipulate this array in parallel to the keyed elements: examples are given in the instructor presentation.

(b) How much additional time and space does your modification require? Justify your response.

> **Solution:** The additional space required is O(n lg n) because we record each of n positions and the largest position is n, which requires ⌈lg n bits⌉. The asymptotic time does not change because modification of the keys into tuples can be done in Θ(n) time and the extra comparison is constant time.
> **Note:** we have asked TAs to give full credit for Θ(n) space, as the point about the representation of n is less obvious.

2. Radix Sort

Use induction to prove that radix sort works. The base case is d=1. Induction will go from lower order to higher order digits. Your proof will need to rely on the assumption that the intermediate sort is stable. *(If it does not, your proof is incomplete! Hint: argue each case where the digits are <, =, and > separately.)*

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

> **Solution:**
> *Base case:* if d=1, sorting on that single digit sorts the array.
> *Induction:* Assume that radix sort works for d-1 digits. We need to show it works for d

digits.

(Radix sorts separately on each digit, starting with the low order digit 1. Radix sort of d digits is equivalent to sorting on the low-order d-1 digits followed by a sort on digit d. By induction the sort on the low order d-1 digits is correct, so the elements are in order according to these digits. The sort on the dth digit will order the elements by this digit. We need to show that this sort is correct for digits d..1)

Consider any two elements x and y with dth digits x_d and y_d.

- If $x_d < y_d$ the d^{th} pass of the sort will put x before y, which is correct, since x < y regardless of the low order digits.
- If $x_d > y_d$ the d^{th} pass of the sort will put y before x, which is correct, since y < x regardless of the low order digits.
- If $x_d = y_d$ the d^{th} pass of the sort will leave x and y in the same order they were in, because it is stable, But that order is already correct as it was determined by the sort on the d-1 digits.

Therefore the sort on the dth digit results in a correct overall sort.

Note: the above three cases do not imply that keys are compared. That would make it a comparison sort, and hence O(n lg n). The cases are for purposes of analysis only, to show that no matter what the relationship is on the d^{th} digit, the result will be correct.