COMP 460 Lecture Notes

R. I. Greenberg

1 Sorting in Linear Time

1.1 Counting Sort

Assumes input elts. $A[1..n]$ are integers in the range 1 to k.

1. $C[i] \leftarrow$ no. of elts. of A that equal i.
   (Do it by marching through input array once and tallying values.)

2. Change $C[i]$ to no. of elts. $\leq i$.
   ($C[i] \leftarrow C[i] + C[i-1]$ for $i = 2, 3, 4, \ldots k$.)

3. Put each elt. of A into correct position of output array B:

   for $j \leftarrow n$ **downto** 1
       $B[C[A[j]]] \leftarrow A[j]$
       $C[A[j]] \leftarrow C[A[j]] - 1$
   endfor

This sort is stable: inputs with same value appear in output array in same order as in input array. (This matters when "satellite data" being carried around with keys being sorted.)

Running time: $O(k+n)$
$= O(n)$ when $k = O(n)$.

1.2 Radix Sort

Sorts nos. digit by digit (or other keys field by field).

Let $d =$ no. of digits in a number
$k =$ max. range of digits (e.g., digits from 1 to k or 0 to $k-1$)

k is called the radix or number base.

Intuitive digit-by-digit approach would be:
Sort by most significant digit; then recursively sort the k collections of nos. having same first digit. Gives a lot of subarrays to keep track of.

Radix sort counterintuitively sorts on least significant digit first:

```
  RADIX-SORT(A, d)
1 for i ← 1 to d
2     Do a stable sort of array A on ith digit from right.
3 endfor
```

If k not too large, counting sort is a good choice for line 2.

Running time: $\Theta(n+k)$ for each execution of line 2.
Total time: $\Theta(d(n+k))$.

When d constant and $k = O(n)$, time for radix sort is $O(n)$. This is often an appropriate perspective, since we tend to build computers with around $\Theta(\lg n)$-bit numbers for the largest n used in practice.

So radix sort time may be good in practice, but it does not sort in place (when based on counting sort).

1.3 Bucket Sort

$\Theta(n)$ average time for random inputs uniformly distributed over interval $[0,1)$, for example.

Divide $[0,1)$ into n equal-sized buckets. Put each elt. in correct bucket. Go through buckets in order, sorting nos. in each bucket by, e.g., insertion sort.

Let n_i be no. of elts. in bucket i for $0 \leq i \leq n-1$. Expected time to do the insertion sorts is

$$\sum_{i=0}^{n-1} \mathrm{E}[O(n_i^2)] = O(\sum_{i=0}^{n-1} \mathrm{E}[n_i^2]) \ ,$$

which the book shows is $O(n)$. The book gives a somewhat lengthy argument using only elementary principles. Here is a shorter argument using some more advanced results that are derived elsewhere in the text. Letting $p = 1/n$ denote the probability of a particular item following into a particular bucket, we have:

$$\begin{aligned}
\mathrm{E}[n_i^2] &= \mathrm{Var}[n_i] + \mathrm{E}^2[n_i] && \text{By Eqn. C.27} \\
&= np(1-p) + (np)^2 && \text{By Eqns. C.37 and C.39} \\
&= 1 - \frac{1}{n} + 1^2 \\
&= 2 - \frac{1}{n} \\
&= \Theta(1) \ .
\end{aligned}$$