Radix Sort

Radix sort is a small method that many people intuitively use when alphabetizing a large list of names. (Here Radix is 26, 26 letters of alphabet). Specifically, the list of names is first sorted according to the first letter of each names, that is, the names are arranged in 26 classes. Intuitively, one might want to sort numbers on their most significant digit. But Radix sort do counter-intuitively by sorting on the least significant digits first. On the first pass entire numbers sort on the least significant digit and combine in a array. Then on the second pass, the entire numbers are sorted again on the second least-significant digits and combine in a array and so on.

Following example shows how Radix sort operates on seven 3-digits number.

INPUT	**1st pass**	**2nd pass**	**3rd pass**
329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

In the above example, the first column is the input. The remaining shows the list after successive sorts on increasingly significant digits position. The code for Radix sort assumes that each element in the n-element array A has d digits, where digit 1 is the lowest-order digit and d is the highest-order digit.

RADIX_SORT (A, d)

```
for $i \leftarrow 1$ to d do
    use a stable sort to sort $A$ on digit $i$
    // counting sort will do the job
```

Analysis

The running time depends on the stable used as an intermediate sorting algorithm. When each digits is in the range 1 to k, and k is not too large, COUNTING_SORT is the obvious choice. In case of counting sort, each pass over n d-digit numbers takes $O(n + k)$ time. There are d passes, so the total time for Radix sort is $\Theta(n+k)$ time. There are d passes, so the total time for Radix sort is $\Theta(dn+kd)$. When d is constant and $k = \Theta(n)$, the Radix sort runs in linear time.

Back