7.7

Sorting on Several Keys

7.7

Sorting with Several Keys

A list of records is said to be sorted with respect to the keys $K^1, K^2, \ldots, K^r$ iff for every pair of records i and $j, i < j$ and

$$(K_i^1, K_i^2, \ldots, K_i^r) \leq (K_j^1, K_j^2, \ldots, K_j^r)$$

$(x_1, \ldots, x_r) \leq (y_1, \ldots, y_r)$
iff either $x_k = y_k, 1 \leq k \leq n$, and
$x_{n+1} < y_{n+1}$ for some $n < r$,
or $x_k = y_k, 1 \leq k \leq r$

Sorting a Deck of Cards

- Each card has two keys
  - K^1 (Suits): ♣ < ♦ < ♥ < ♠
  - K^2 (Face values): 2 < 3 < 4 …<J < Q < K < A
  - The sorted list is: 2 ♣, …,A♣, …, 2 ♠, …,A ♠
- Most-significant-digit (**MSD**) sort
  - Sort using K^1 to obtain 4 "piles" of records.
  - Sort each piles into sub-piles.
  - Merge piles by placing the piles on top of each other.

Sorting a Deck of Cards (cont'd)

- Least-significant-digit (**LSD**) sort
  - Sort using K^2 to obtain 13 "piles" of records.
  - Place 3's on top of 2's,…,Aces on top of kings.
    - 2 < 3 < 4 … J < Q < K < A
  - Using a stable sort with respect to K^1 and obtain 4 "piles".
  - Merge piles by placing the piles on top of each other.

71

Bin Sort (Bucket Sort)

- Assume the records in a list to be sorted come from a set of size $\boldsymbol{m}$, say $\{1,2,\ldots,m\}$.
- Create $\boldsymbol{m}$ buckets.
- Scan the sequence $a[1]$ … $a[n]$, and put $a[i]$ element into the $\boldsymbol{a[i]^{th}}$ bucket.
- Concatenate all buckets to get the sorted list.
- Suitable for a set with small $\boldsymbol{m}$.

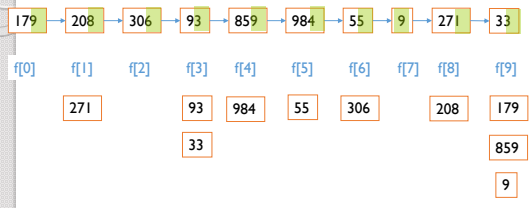
72

Radix Sort

- Decompose the key (number) into subkeys using some **radix** $\boldsymbol{r}$
  - For $r = 10, K = 123$, then $K^1 = 1, K^2 = 2,$ and $K^3 = 3$.
- Create $\boldsymbol{r}$ buckets (0 ~ $\boldsymbol{r-1}$).
- Apply bin sort with MSD or LSD order.
- Suitable to sort numbers with large value range.

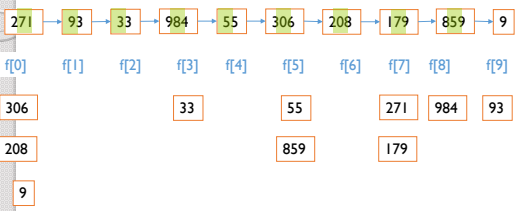
73

Radix Sort Example (Pass 1)



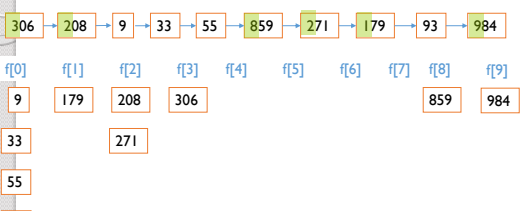
74

Radix Sort Example (Pass 2)



75

Radix Sort Example (Pass 3)



Time Complexity: O(d*(n+r))

76

LSB Radix Sort (code) 1/2

```
template <class T>
int RadixSort(T *a, int *link, const int d, const int r, const int n)
{// using a radix sort with d digits`radix r to sort a[1:n]
 // digit(a[i], j, r) return the j-th key in radix r of a[i]
 // each digit is within the range [0, r). Using the bin sort to
 // sort elements of the same digit.
 int e[r], f[r]; // head and tail of the bin
 int first = 1; // start from the 1st element
 for(int i =1; i < n; i++) link[i]=i+1; // link the elements
 link[n] = 0;
 // do radix sorting…
```

77

LSB Radix Sort (code) 2/2

```
 // do radix sorting…
 for (i = d-1; i >=0; i--) { // sort in LSB order
   fill(f, f+r, 0); // initialize the bins
   for (int current = first; current; current = link[current])
   { // put the element with key k to bin[k]
     int k = digit(a[current], i, r);
     if (f[k]== 0) f[k] = current;
     else link[e[k]] = current;
     e[k] =current;
   }
   for (j = 0; !f[j]; j++); // find the 1st non-empty bin
   first = f [j];
   int last = e[j];
   for (int k = j + 1; k < r; k++){ // link the rest of bins
     if (f[k]) {
        link[last] = f[k];
        last = e[k];}
   }
   link[last] = 0;
 }
 return first;
}
```

78