Digital Sorting

- *Comparison* sorting uses only $<, >, =$
- *Digital* sorting takes advantage of item representation.
  - strings are sequences of characters
  - numbers are sequences of bits
  - sorting key comes from a limited range

Sorting Students by Grade
Sorting Students by Grade

Could use one of our sorting algorithms

Bucket Sort

Easier to just group by grade

Bucket Sort

```java
// elements in array have key values 0...N-1
void BucketSort(Elt array, int n)
{
    for (i = 0 to N-1) Q[i] = new EltQueue;
    for (j = 0 to n-1)
        Q[array[j]. key]. enqueue(array[j]);
    j = 0;
    for (i = 0 to N-1) {
        while (!Q[i]. isEmpty()) {
            array[j] = Q[i]. dequeue();
            j++;
        }
    }
}"
```
Bucket Sort

We use queues to keep students alphabetized by name within each grade

(i.e. with queues, bucket sort is stable)

Bucket Sort Analysis

Putting the Digit in Digital

Suppose we want to sort a bunch of 10-digit numbers

- Range too large for bucket sort
- But each digit has small range.
- Can we iterate bucket sort?
Radix Sort

```java
// Integer is a class for our D-digit numbers
void RadixSort(Integer[] array, int n)
{
    for (i = 0 to 9) Q[i] = new IntegerQueue;
    for (k = 0 to D-1)
        for (j = 0 to n-1)
            Q[array[j].Digit[k]].enqueue(array[j]);
    for (i = 0 to 9)
        while (!Q[i].isEmpty())
            array[j] = Q[i].dequeue();
}
```

This works for the same reason bucket sort on grades keeps names alphabetized: **stability**

Radix Sort Example

```
259 168 249 368 287
  7  8  9
247 158 368 259 249
  4  5  6
247 249 158 259 368
  1  2  3
158 247 249 259 368
```

Radix Sort Analysis

Assume \( n \) items of \( D \) “digits” of range \( 0 \ldots N – 1 \).
Radix Sort: What It Looks Like

1-bit Radix

1. Initial array
2. First bit
3. Second bit
4. Third bit
5. Finished

Extending Radix Sort

- Don’t need to use decimal digits
  - binary, ASCII (8-bit), ...
- Doesn’t need to be numbers
  - Strings. Variable-length?

Radix Sort Question

Why do we sort from right to left and not left to right?
Radix Exchange Sort

// sort rg[low...high-1] on bits k...0
void RadixExchangeSort(int *rg, int low, int high, int k)
{
    if (k >= 0 && low < high - 1) {
        i = low; j = high-1;
        while (i < j) {
            while (i < j && Bit(rg[i], k) == 0) i++;
            while (i < j && Bit(rg[j], k) == 1) j--;
            if (i < j)
                swap(rg[i], rg[j]), i++, j--;
        }
        if (Bit(rg[i], k) == 0) i++;
        RadixExchangeSort(rg, low, i, k-1);
        RadixExchangeSort(rg, i, high, k-1);
    }
}

A Digital Version of Quicksort

Radix Exchange Sort: What it Looks Like

Radix Exchange vs. Quicksort
Lazy Radix Exchange

- Array mostly sorted by last few bits
- Is there an easier way to finish a mostly sorted array?

stop at bit 4  stop at bit 3
stop at bit 2  stop at bit 1