RADIX & BUCKET SORT
Algorithms, Complexity

Sorting in Better Than $\Omega(N \log N)$ Time?

- Comparison-based sorting algorithms:
  - Use only comparisons between elements for sorting
  - Require $\Omega(N \log N)$ worst-case comparisons (Section 7.8)

- You can still sort in linear time in some special cases:
  - e.g., Bucket sort, Radix sort
  - But, extra information must be available for each to work
Bucket Sort: Algorithm and Example

- If input consists of only positive integers less than M:
  - Initialize an array, called `count`, of size M to all 0's
    - Count has M buckets, which are initially empty
  - When $A_i$ is read, increment `count[A_i]` by 1
  - After all input is read
    - scan `count`, printing out a representation of the sorted list

```
3 8 5 2 9 2
```

```
0 2 1 0 1 0 0 1 1 0
```

```
2 2 3 5 8 9
```

Bucket Sort: Properties & Limitation

- Run-time:
  - $O(M+N)$
  - If M is $O(N)$, then run-time is $O(N)$

- How does it improve the comparison-sort lower bound?
  - Uses a more powerful operation than simple comparisons
  - Performs M-way comparisons in unit time!

- Limited usability if values can be large (< M)
  - Too many buckets needed!
  - e.g., just 10 numbers in the range 0 to 999!
Radix Sort: Algorithm and Example

- Radix sort: if input has $N$ numbers in the range $[0, b^p-1]$
  - Use several passes of bucket sort
  - Assuming each number is expressed to base $b$,
    - Bucket sort on the least significant “digit”
    - Then on the next least significant digit
    - ...
  - Numbers in a given bucket could be different
    - So keep a list, not simply count

- Example input: 64, 8, 216, 512, 27, 729, 0, 1, 343, 125

Radix Sort: Properties

- Each pass is stable
  - Items that agree in the current digit retain the ordering
determined in prior passes
  - This is why the algorithm works in the first place!

- Run-time
  - $O(p(N+b))$
    - $p$: number of passes
    - $N$: number of elements to sort
    - $b$: number of buckets
Radix/Bucket Sort: Applications

- Sorting strings:
  - If all strings have same length L, radix sort is O(NL)

- Resource scheduling
  - e.g., CPU, network access, disk
  - If two processes have same deadline, can assign them to the same “bucket”
    - Linear-time scheduling, rather than quadratic-time

In-class Problems

- Suppose you have an array of N elements containing only two distinct keys: true and false.
  - Give an O(N) algorithm to rearrange the list so that all false elements precede the true elements.
  - You may use only constant extra space.
Suppose you are given a sorted list of N elements followed by f(N) randomly-order elements

- How would you sort the entire list if f(N) = O(1)?
- How would you sort the entire list if f(N) = O(log N)?
- How large can f(N) be for the entire list still to be sortable in O(N) time?

Write (and solve) a recurrence relation for binary search