More on Sorting

CSE 2011

Winter 2011

When to use which sorting algorithm?

- Large arrays: merge sort, quick sort.
- Small arrays: insertion sort, selection sort.
  - Recursion is expensive.
- Merge sort or quick sort in an average case?
  - Cost of comparing elements
  - Cost of moving/switching elements

Merge Sort or Quick Sort?

Merge sort
- Lowest number of comparisons among popular algorithms
- Lots of data movements/copying (merging)

Java
- Generic sort uses Comparator

⇒ comparison is expensive.
- Moving is cheap (uses "pointers" rather than copies of objects).

Quick sort
- More comparisons
- Fewer data movements

C++
- Copying large objects is expensive.
- Comparison is cheap (compiler does inline optimization).

Java
- Used for primitive types (inexpensive comparisons)

3

Lower Bound for Sorting

- Merge sort and heap sort (discussed later)
  - worst-case running time is O(N log N)
- Are there better algorithms? No.
- We need to prove that any sorting algorithm based on only comparisons takes Ω(N log N) comparisons in the worst case (worse-case input) to sort N elements.
- We will prove this after learning "Trees".

4

Linear Time Sorting (O(N))

CSE 2011

5

Linear Time Sorting

- Can we do better (linear time algorithm) if the input has special structure (e.g., uniformly distributed, every number can be represented by d digits)? Yes.
- Counting sort, radix sort, bucket sort

6

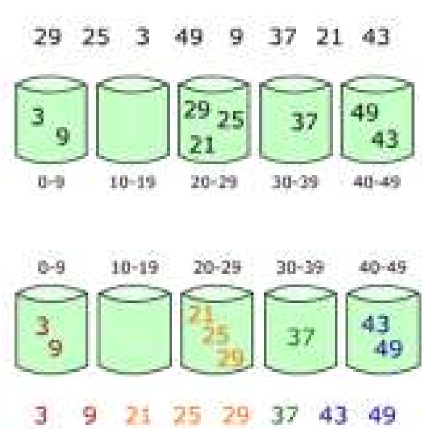
Bucket Sort

- Given an integer array A of size N,
- Assume that all elements in A have values < ***m***.
- Example: sort a list of students by grades: 9 (A+), 8 (A), 7 (B+), 6 (B), 5, 4, 3, 2, 1, 0.
- Create an array B of size M. Each entry B[i] is considered a "bucket".
- For each element A[i], "throw" the element into bucket B[A[i]].

7

Bucket Sort: Example

- Each bucket contains more than one key values.
- After all inputs are thrown into the buckets, each bucket will be sorted (e.g., using insertion sort).



8

Bucket Sort: Running Time

- Assume there are m buckets.
- Assume uniform distribution of elements into buckets.
- Then the bucket size is $k \cong N / m$.
- Algorithm:
  - Create m buckets.
  - “Throw” N elements into the appropriate buckets.
  - Insertion sort each bucket.
  - Concatenate the sorted lists.

9

Bucket Sort: Running Time (2)

- Algorithm:
  - Create m buckets $\Rightarrow O(m)$
  - “Throw” N elements into the buckets $\Rightarrow O(N)$
  - Insertion sort each bucket $\Rightarrow O(k^2) \times m = O(N^2 / m)$
  - Concatenate the sorted lists $\Rightarrow O(N)$

- Total = $O(N^2 / m) + O(N) + O(m)$

- If $m = \Theta(N)$, e.g., $m = N/100$, then the running time of bucket sort is $O(N)$.

10

Next time …

- Arrays (review)
- Linked Lists (3.2, 3.3)