Other N log N Sorts

- **Binary Tree Sort**
  - Basic Recipe
    - Insert into binary search tree (BST)
    - Do Inorder Traversal
  - Complexity
    - Create: $O(N \log N)$
    - Traversal $O(N)$
  - Not usually used for sorting unless you need BST for other reasons
Other $N \log N$ Sorts

- **Heap Sort**
  - Basic recipe:
    - Create Heap (priority queue)
    - Get items one at a time (Sorted order!)
  - Complexity
    - Create heap: $N \cdot O(1) = O(N)$
    - Remove $N$ items: $N \cdot O(\log N) = O(N \log N)$
  - To make into sort:
    - Use Max-Heap on array
    - Put removed items *into space vacated as heap shrinks*
    - Thus sort "in place": no extra array needed
  - Not widely used sort; not stable
Shellsort

- Uses Insertion Sorts with gaps (or skips)
  - “Diminishing Gap Sort” (Donald Shell, 1959)
  - Gap = 5 (5 insertion sorts with every 5th element)
  - Gap = 3 (3 insertion sorts with every 3rd element)
  - Gap = 1 (standard insertions sort)

- Complexity
  - Very hard to analyze: depends on gaps used
  - $O(N^{3/2})$ fairly easy to achieve; can do better
  - Easy to program
Non-comparison-based sorts

- **Lower bound:** $\Omega(n \log n)$ for comparison-based sorts (like searching lower bound)
- **Bucket sort/radix sort** are not comparison based, faster asymptotically and in practice
- **Sort a vector of ints, all ints in the range 1..100, how?**
  - (use extra storage)
- **Radix:** examine each digit of numbers being sorted
  - One-pass per digit
  - Sort based on digit
  - What order should passes be in?

```
23 34 56 25 44 73 42 26 10 16
16
26
10 42 23 73 34 44 25 56 26 16
```
External Sorting

- Large memories on modern machines means techniques discussed so far usually apply
- Sometimes data does not fit into memory
  - This used to be a common data processing problem
- Usual Recipe:
  - Chop data into chunks that will fit into memory
  - Sort chunks in memory using best programs
    - Use Quicksort for speed, or Merge Sort for stable sort
    - Write sorted chunks back to disk files
  - Merge the disk files
    - Read front of 2 or more files
    - Merge
    - Write to final disk file as you merge
  - Only small part needs to be in memory at any time
- Historically all done with tapes (disks too small)