**Sorting**

**Radix**

Radix sorting algorithm is quite different from the others. It uses the idea of forming groups and then combining them to sort a collection of data. As an example of radix sort, consider this collection of three-letter strings:  
ABC, XYZ, BWZ, AAC, RLT, JBX, RDT, KLT, AEO, TLJ  
  
**Group strings by rightmost letter**  
The sort begins by organizing the data, according to their rightmost(least significant) letters. Two strings end in C. Place those two strings into a group. Continuing through the alphabet you form the following groups:  
(ABC, AAC) (TLJ) (AEO) (RLT, RDT, KLT) (JBX) (XYZ, BWZ)  
The strings in each group end with the same letter, and the groups are ordered by that letter.  
  
**Combine groups**  
Now combine the groups into one as follows. Take the item in the first group in their present order, follow them with the items in the second group in their present order, and so on. The following group results:  
ABC, AAC, TLJ, AEO, RLT, RDT, KLT, JBX, XYZ, BWZ

**Group strings by middle letter**Next form new groups as you did before, but this time use the middle letter of each string instead of the last letter.  
(AAC), (ABC, JBX), (RDT), (AEO), (TLI, RLT, KLT), (BWZ), (XYZ)  
Now the strings in each group have the same middle letter, and the groups are ordered by that letter.   
  
**Combine groups**Combine these groups into one group, again preserving the relative order of the items within each group:  
AAC, ABC, JBX, RDT, AEO, TLI, RLT, KLT, BWZ, XYZ  
  
**Group strings by first letter**Now from a groups according to the first letter of each string:  
(AAC, ABC, AEO), (BWZ), (JBX), (KLT), (RDT, RLT), (TLI), (XYZ)  
  
**Sorted strings**  
Finally, combine the groups, again maintaining the relative order within each group:  
 AAC, ABC, AEO, BWZ, JBX, KLT, RDT, RLT, TLI, XYZ  
The strings are now in sorted order.

In the previous example, all character strings had the same length. If the character strings have varying lengths, you can treat them as if they were the same length by padding them on the right with blanks as necessary.  
To sort numeric data, the radix sort treats a number as a character string. You can treat numbers as if they were padding on the left with zeros, making them all appear to be the same length:  
0123, 2154, 0222, 0004, 0283, 1560, 1061, 2150  
(1560, 2150), (1061), (0222), (0123, 0283), (2154, 0004)  
1560, 2150, 1061, 0222, 0123, 0283, 2154, 0004  
(0004), (0222, 0123), (2150, 2154), (1560,1061), (0283)  
0004, 0222, 0123, 2150, 2154, 1560, 1061, 0283  
(0004, 1061), (0123, 2150, 2154), (0222, 0283), (1560)  
0004, 1061, 0123, 2150, 2154, 0222, 0283, 1560  
(0004, 0123, 0222, 0283), (1061, 1560), (2150, 2154)  
0004, 0123, 0222, 0283, 1061, 1560, 2150, 2154

You can see that this algorithm requires *n* moves each time if forms groups and *n* moves to combine them again into one group. The algorithm performs these *2\*n* moves *d* times. Therefore, the radix sort requires *2\*n\*d* moves to sort *n* strings of *d* character each. However, notice that no comparisons are necessary. Thus, radix sort is *O(n).*

References  
[1] Carrano Prichard Data Abstraction and Problem Solving with Java 3rd ed.