**Radix sort**

Definition: Radix

MATHEMATICS: the base of a system of numeration.

FORMAL: a source or origin of something. "Judaism is the radix of Christianity"

**Radix sort** is a non-[comparative](http://en.wikipedia.org/wiki/Comparison_sort) [integer](http://en.wikipedia.org/wiki/Integer_sorting) [sorting algorithm](http://en.wikipedia.org/wiki/Sorting_algorithm) that sorts data with integer keys by grouping keys by the individual digits which share the same [significant](http://en.wikipedia.org/wiki/Significant_figures) position and value. A [positional notation](http://en.wikipedia.org/wiki/Positional_notation) is required, but because integers can represent strings of characters (e.g., names or dates) and specially formatted floating point numbers, [radix](http://en.wikipedia.org/wiki/Radix) sort is not limited to integers. Radix sort's time complexity is O(kn) where n is the number of keys to be sorted and k is the key length

Two classifications of radix sorts are [least significant digit](http://en.wikipedia.org/wiki/Least_significant_digit) (LSD) radix sorts and [most significant digit](http://en.wikipedia.org/wiki/Most_significant_digit) (MSD) radix sorts. LSD radix sorts process the integer representations starting from the least digit and moves towards the most significant digit. The most significant digit starts from the MSD and moves towards the LSD. The examples below implement the former classification.

Original, unsorted list:

170, 45, 75, 90, 802, 2, 24, 66

Sorting by least significant digit (1s place) gives:

170, 90, 802, 02, 24, 45, 75, 66

Notice that we keep 802 before 2, because 802 occurred before 2 in the original list, and similarly for pairs 170 & 90 and 45 & 75.

Sorting by next digit (10s place) gives:

802, 002, 024, 045, 066, 170, 075, 090

Notice that 802 again comes before 2 as 802 comes before 2 in the previous list.

Sorting by most significant digit (100s place) gives:

002, 024, 045, 066, 075,090, 170, 802

It is important to realize that each of the above steps requires just a single pass over the data, since each item can be placed in its correct bucket without having to be compared with other items.

EXAMPLE:

**Original, unsorted list:**

**[170, 45, 75, 90, 2, 802, 24, 66]**

**1ST PASS:**

The integers are enqueued into an array of ten separate queues based on their digits from right to left.

**0:** 170, 090

**1:** none

**2:** 002, 802

**3:** none

**4:** 024

**5:** 045, 075

**6:** 066

**7: none**

**8: none**

**9**: none

The queues are dequeued back into the array of integers, in increasing order.

**[170, 090, 002, 802, 024, 045, 075, 066]**

**2nd PASS**

**0:** 002, 802

**1:** none

**2:** 024

**3:** none

**4:** 045

**5:** none

**6:** 066

**7:** 170, 075

**8:** none

**9:** 090

The queues are dequeued back into the array of integers, in increasing order.

**[002, 802, 024, 045, 066, 170, 075, 090]**

**3rd PASS**

Queues:

**0:**002, 024, 045, 066, 075, 090

**1:**170

**2:** none

**3:** none

**4:** none

**5:** none

**6:** none

**7:** none

**8:**802

**9:** none

The queues are dequeued back into the array of integers, in increasing order.

**[002, 024, 045, 066, 075, 090, 170, 802]** (sorted)

**Algorithm**

* Find the largest element in the array (MAX)
* M=10, N=1 (Used to find the least significant digit, start with the right most digit)
* Create a 10 element array of queues.
* Continue while there are more significant digits that have to be processed (N<=MAX). For each pass process one of the significant digits
* Identify the least significant digits starting from the right most digit using mod and division and place in appropriate queue.

For example: 195 is going to be placed in one of the queues based on LSD

195%10 = 5

5/1=**5** 1st least significant digit (M=10, N=1) place in queue 5

195%100=95

95/10=**9** 2nd least significant digit (M=100, N=10) Place in queue 9

195%1000=195

195/100=**1** 3rd least significant digit (M=1000, N=100) place in queue 1

* Dequeue all the queues and Repopulate array
* M=M\*10, N=N\*10 (Process the next least significant digit)

Hint:

**Bad**

node \*front0, \*rear0;

node \*front1, \*rear1;

node \*front2, \*rear2;

…

enqueue(front0,rear0,data);

enqueue(front1,rear1,data);

….

**Better**

node \*front[10], \*rear[10];

enqueue(front[i],rear[i],data);

**Best**

struct node{

int data;

node \*next;

};

struct nodeQ{

node \*front;

node \*rear;

};

nodeQ que[10];

enqueue(que[i],data);