“Radix Sort”

Best way to sort physical objects

• “algorithm” used to sort punched cards by hand or by machine since 1890
Thinking some more

Radix sort is good for cards.
  • Sorting cards involves moving them around.
    Easy to move cards around in stacks.

Not so good in computer memory.
  • Sorting involves copying data.
    Each data item must be copied individually.

Sorting “in place”

Fastest computer algorithms avoid copying data
  • unless it’s specifically out of order
**Idea - quicksort**

Reorder a list of items into two parts
- all the items in the first part are small
- all the items in the second part are big
- move as few items as possible

Then sort each part separately

---

**Specific idea**

Start with your list of numbers:

211, 220, 012, 201, 210, 112, 002, 020, 120, 101

Pick one of them at random
- here last, 101
- could also use middle one
- or random number generator
Specific idea

Your list of numbers:
211, 220, 012, 201, 210, 112, 002, 020, 120, 101

Scan through the numbers by 101
- front to back and back to front simultaneously
- leaving bigger numbers in back
- leaving smaller numbers in front
- swapping numbers when necessary

Specific idea

Scan your list of numbers:
211, 220, 012, 201, 210, 112, 002, 020, 120, 101

First few steps:
- 211 is bigger than 101, need to swap
- 120 is bigger, can leave in place
- 020 is smaller than 101, need to swap
Scan your list of numbers:
020, 220, 012, 201, 210, 112, 002, 211, 120, 101

Next few steps:
• swap and continue
• 220 needs to swap
• 002 needs to swap

Scan your list of numbers:
020, 002, 012, 201, 210, 112, 220, 211, 120, 101

Next few steps:
• swap and continue
• 012 is OK, 201 needs to swap
• 112, 210, 201 are all OK
• That’s end of scan
**Specific idea**

Swap 201 and 101 to get two ordered parts:

020, 002, 012, 101, 210, 112, 220, 211, 120, 201

Now sort in place:

020, 002, 012

And sort in place:

210, 112, 220, 211, 120, 201

**Recursion!**

Sort this the same way:

020, 002, 012 → 002, 012, 020

Pick 012 and scan from left and right.

Swap 002 and 020. Complete scan.

Swap 020 and 012

No more recursion needed
Recursion!

Sort this the same way:

210, 112, 220, 211, 120, 201

Pick 201 and scan from left and right.

Left needs to swap 210.
Right needs to swap 120.
Swap and continue

Recursion!

Sort this the same way:

120, 112, 220, 211, 210, 201

Pick 201 and scan from left and right.

112 is OK, 220 needs to swap.
211 is OK, 220 completes the scan.
Swap 220 and 201
Recursion!

Sort this the same way:

120, 112, 201, 211, 210, 220

Sort two sublists:

120, 112 → 112, 120 (one step)
211, 210, 220 → 210, 211, 220 (two steps)

Recursion!

Final result:

002, 012, 020, 101, 112, 120, 201, 210, 211, 220
Code for scan (fussy)

Assume item RIGHT of L is the pivot
Set i = LEFT - 1
Set j = RIGHT
repeat until not (j > i)
  • change i by 1
  • repeat until not (item i of L < item RIGHT of L)
    • change i by 1
  • change j by -1
  • repeat until not (item j of L > item RIGHT of L and j > i)
    • change j by -1
  • if (i < j)
    • SWAP i and j in L
SWAP i and RIGHT in L.

Consequence

Each scan takes at most N steps.
As long as we divide list roughly in half, takes NlogN time – same as always!
Unless we’re unlucky (list almost sorted), it turns out, this will divide list roughly in half.
Euclid’s algorithm

– suppose we want to tile this with square tiles

Euclid’s algorithm

– know there’s a tile boundary at y – draw that in
Euclid’s algorithm

\[ x \mod y \]

– have to be able to tile this area with same tiles!

Euclid’s algorithm

\[ y \mod z \]

\[ z = x \mod y \]

– repeat! - until tile fits exactly
Euclid’s algorithm

$z = x \mod y$

- repeat! - until tile fits exactly

$y \mod z$

Euclid’s algorithm

$z = x \mod y$

- repeat! - until tile fits exactly

$y \mod z$