CMSC351-2023-10-25

Wednesday, October 25, 2023 8:24 AM

Counting Sort, Continued

(1) Recall: Counting sort works for integers b/w 0, k (inclusive). (think: k = maximum!)
It's stable, not in-place, and has time complexity $\Theta(n+k)$.

(2) Other Notes

(A) Best used when k is indep. of n b/c then we treat it as constant and we have time $\Theta(n)$.
ex Sorting lists of length n where all entries are b/w 0, 423 (inclusive)
this is $\Theta(n+423) = \Theta(n)$

(B) Doesn't have to be!
ex Sorting lists of length n where all entries are b/w 0, n^2 (inclusive)
this is $\Theta(n+n^2) = \Theta(n^2)$
In such a case we might prefer a different algorithm!

(C) Aux Space: $\Theta(n+k)$ b/c ANEW is length n
and POS is length k+1
and some constants.

(D) This can make C. Sort wasteful!
ex Sorting lists of length n where all entries are b/w 1M and 1M+3.
we'd have aux space $\Theta(n+1M+3) = \Theta(n)$ ← lots of wasted space!

(E) Counting sort can be modified sometimes!
ex Sort lots of fractions of the form $\frac{1}{a}$ w/ $a \in \mathbb{Z}^+$. e.g. $[\frac{1}{5}, \frac{1}{10}, \frac{1}{100}, \frac{1}{2}, \dots]$
we could invert all entries: $[5, 10, 100, 2, \dots]$
then counting sort $[2, 5, 10, 100, \dots]$
re-invert $[\frac{1}{2}, \frac{1}{5}, \frac{1}{10}, \frac{1}{100}, \dots]$
reverse $[\dots, \frac{1}{100}, \frac{1}{10}, \frac{1}{5}, \frac{1}{2}]$
ex Sort list of int. b/w −10 and 10, inclusive.
we could add 10 to each, sort, then subtract 10 from each.
Sometimes not!

we could add 10 to each, sort, then subtract 10 from each.
Sometimes not!
ex Sort list of real numbers.

Radix Sort

① Intro: Designed to handle values or strings or expressions which can be written in a certain base/radix.
Ideally w/ a max/fixed # of digits.
ex Sort decimal numbers b/w 000 and 999. ← base = 10
ex Sort binary numbers b/w 00000000 and 11111111 ← base = 2
ex Sort strings of 5 letters each, A-Z. So like [HAPPY, HELLO, MEATY, ...]
↖ base = 26

② Method:
First stable sort by least significant digit
then by next least sig. dig.
etc.
⊗ technically any stable sorting alg. will do, but... ✓
ex A = [123, 311, 213, 141, 222, 321]

A = [311, 141, 321, 222, 123, 213]

A = [311, 213, 321, 222, 123, 141]

A = [123, 141, 213, 222, 311, 321] ← sorted!

note 1: tempting to think we should do most sig. dig first, but that fails!
note 2: Can be done w/ most sig. dig. first but we'd need to be more recursive about it.

③ Comments
(1) All expressions must have the same # of digits.
For integers we can left-pad w/ 0's.

(A) ... expressions must have the same # of digits.
For integers we can left-pad w/ 0's.
For strings, ?

(B) What are we doing?
We're doing d separate sorts, where d = # of digits.
Each is with a list of length n.
In each sep. sort, each is actually sorting n single digits
bc the other digits are just going along for the ride.
Sps b = base.
ex Sps we sort n ints. blw 000 and 999. We have:
$n = n$
$d = 3$
$b = 10$
In this ex, each sep. sort is of n items each in the range 0 to b-1.

(C) Since b doesn't dep. on n the obvious choice
for our stable sort is Counting sort. ☺
In such a case each counting sort is $\Theta(n+(b-1))$
Then bc we're doing this d times:
$$\Theta(d(n+(b-1)))$$
Assuming b = fixed, this is $\Theta(dn)$ time complexity

(D) Probably d does not dep. on n so its $\Theta(n)$.

(E) But it could...

ex Lets sort a list of n nonzero int blw 0 and 2^n-1
each represented in binary. What's time complexity?
Soln It takes n binary digits to rep. int. blw $0, 2^n-1$
thus $d = n$ so then time = $\Theta(n \cdot n) = \Theta(n^2)$

(F) Aux Space: Assuming Counting sort, $\Theta(n+(b-1))$
Why not $\Theta(d(n+(b-1)))$?
bc sorts happen in series, mem. freed!

(G) Stable: Yes

b/c sorts happen in series, mem. freed!

(G) Stable: Yes.

(H) IP: No, assuming counting sort!