CMSC351-2023-10-23

Monday, October 23, 2023 8:19 AM

COUNTING SORT

① Intro - what is Counting sort?

A non-comparison-based sorting algorithm which can often have worst-case time complexity better than $\Omega(n \lg n)$.

recall: A CBSA cannot have worst-case better than $\Omega(n \lg n)$.

② An intro example

Consider the list A = | 3 | 0 | 2 | 1 | 2̃ | 0̃ |

First we observe that A contains integers blw 0 and 3. Let k=3. (k = max int. value). Creat a list POS which is length k+1 (indexed from 0 to k). Calculate and assign:

POS[i] = # of i-values in A

so POS = [2, 1, 2, 1]

2 0's in A; 1 1 in A; 2 2's in A; 1 3 in A

then make POS comulative: Starting at POS[1], add the previous value to each element.

so POS = [2, 3, 5, 6]

So now we have

A = | 3 | 0 | 2 | 1 | 2̃ | 0̃ |

POS = | 2 | 3 | 5 | 6 |

the goal is sorted A:

A = | 0 | 0̃ | 1 | 2 | 2̃ | 3 |

observe: For each i blw 0 and k,

POS[i] = last position of i values in sorted version of A.

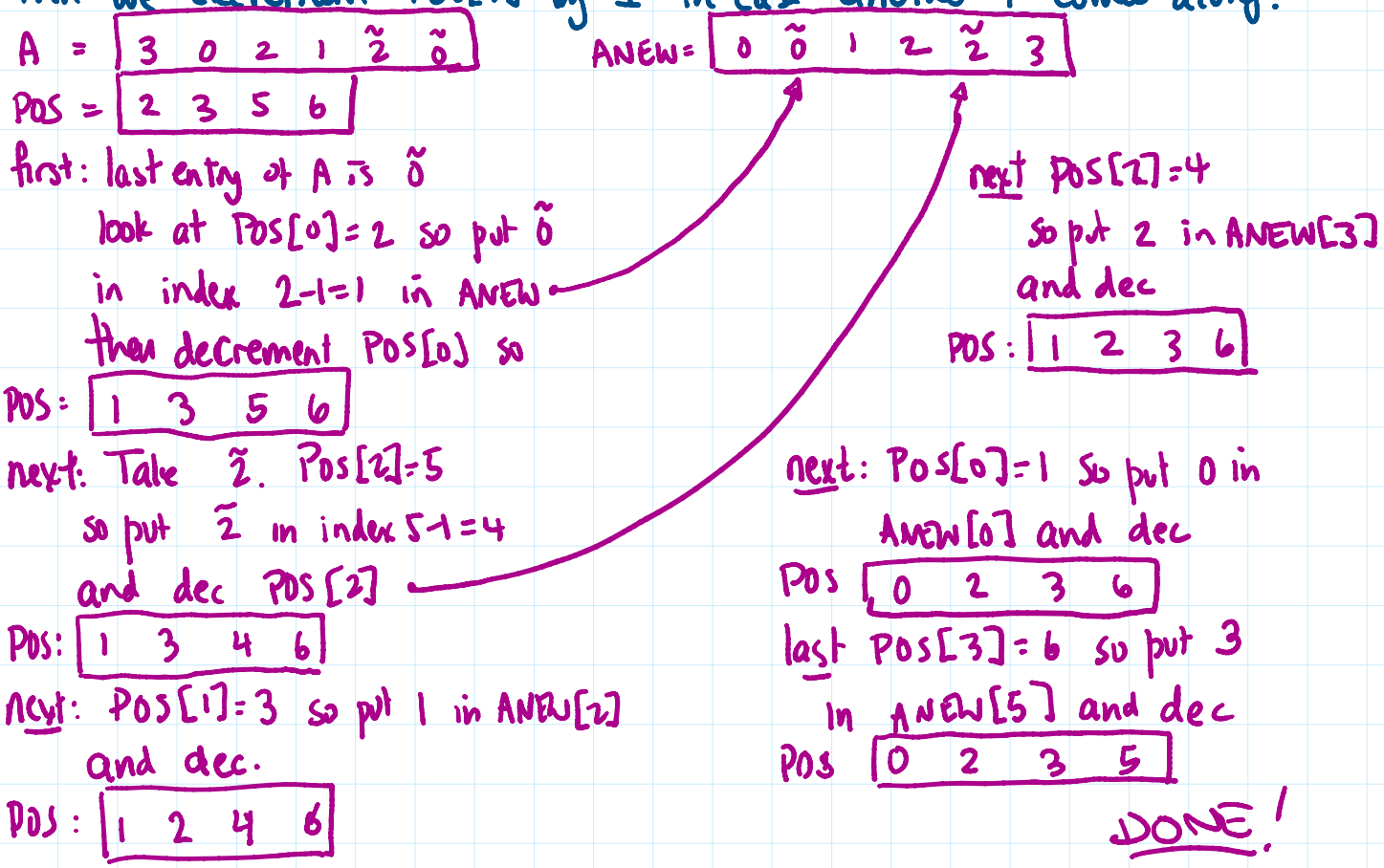
in our ex, POS[0] = 2 → our 0's end at position 2 (index 1)

POS[1] = 3 → our 1's end at position 3 (index 2)

⋮

Thus we create a new list ANEW of same length as A and

Thus we create a new list ANEW of same length as A and fill it in using POS. How?
Go through A in reverse order. For each value i we look at POS[i] to tell us which position it should go in, in ANEW. We assign ANEW[POS[i]-1] = i
then we decrement POS[i] by 1 in case another i comes along!



③ Pre-Pseudocode notes

(A) We go through A backwards blc it makes Counting sort stable.
ex w/ our 0, 0̃ and 2, 2̃.

(B) If all we wanted was a list of integers we could just use our orginal POS to create it.
Think: our POS is giving us guidance/instructions as to how to move the values in A into ANEW in their correct positions!

as to how to move the values in A and ANEW to their correct positions!

(C) Note: if max = k = BIG then POS = BIG! Lots of aux space!

(D) We need to know k, or compute it!
(E) The entries must be nonnegative integers. } Serious restrictions! ∴

④ Pseudocode!

```
\\ PRE: A is a list of length n with minimum 0 and maximum k
POS = list of zeros of length k+1
ANEW = list of zeros of length n
for i = 0 to n-1
    POS[A[i]] = POS[A[i]] + 1
end
for i = 1 to k
    POS[i] = POS[i] + POS[i-1]
end
for i = n-1 down to 0
    ANEW[POS[A[i]]-1] = A[i]
    POS[A[i]] = POS[A[i]] - 1
end
for i = 0 to n-1
    A[i] = ANEW[i]
end
\\ POST: A is sorted.
```

← POS now contains counts of each value
← POS is now cumulative + gives positions!
← copies A[i] to correct index of ANEW
← decrement POS appropriately!
← copy back to A.

⑤ Time and Space

Time Complexity: It's typical that allocating a list of length j is $\Theta(j)$
So our initialization of ANEW is $\Theta(n)$ and of POS is $\Theta(k)$.
The four loops are then $\Theta(n), \Theta(k), \Theta(n), \Theta(n)$.
All together, all cases, it's:

$$T(n,k) = \Theta(n) + \Theta(k)$$

We'll write this as:

$$T(n,k) = \Theta(n+k)$$

Means: $\exists j_0, B, C$ s.t. if $n+k \geq j_0$ then $B(n+k) \leq T(n,k) \leq C(n+k)$

note: if k = const. and doesn't dep. on n
then we get $\Theta(n+k) = \Theta(n)$.

note: Perhaps k dep. on n... for ex if our list of length

note: Perhaps k dep. on n... for ex if our list of length n contains integers blw 0 and n^2 then we get $\theta(n+k) = \theta(n+n^2) = \theta(n^2)$

for ex if blw 0 and $\sqrt{n}$ then $\theta(n+k) = \theta(n+\sqrt{n}) = \theta(n)$