LEC5 Linear Sorting

https://github.com/GUMI-21/MIT6.006_note

3.19

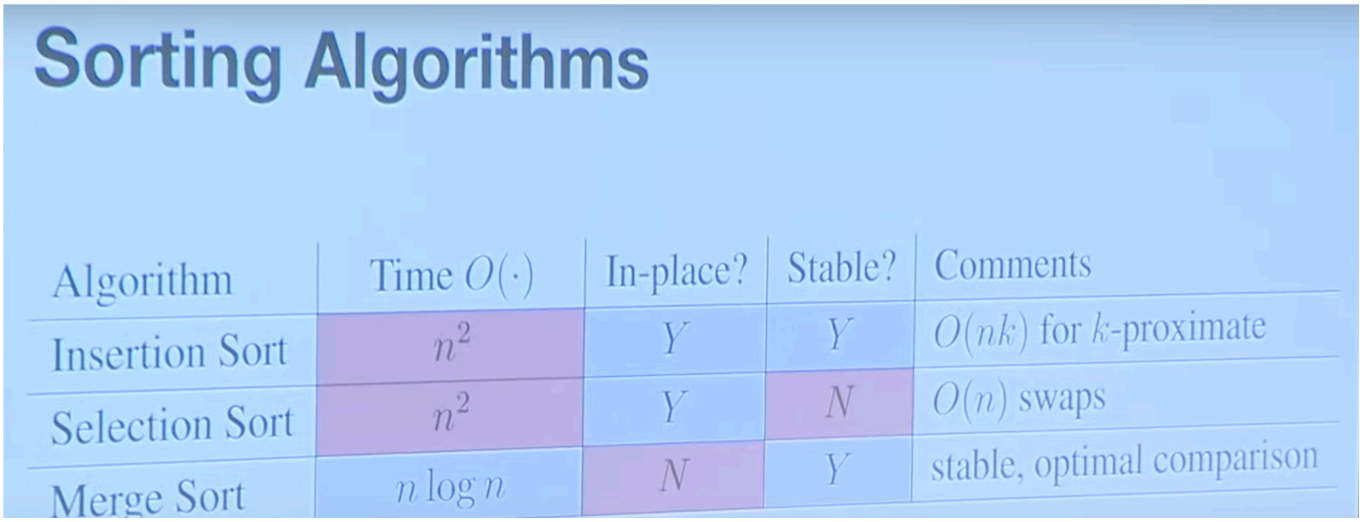
Last Lec Review

- Comparison model
  $\Omega(\log(n))$ time to search
- Do faster using RAM and direct access array
- Soace O(u), reduce space via hash $h(k) : U -> N$
- Expected O(1) time dictionary ops.

Data Structure	Operations $O(\cdot)$				
	Container	Static	Dynamic	Order	
	`build(X)`	`find(k)`	`insert(x)` `delete(k)`	`find_min()` `find_max()`	`find_prev(k)` `find_next(k)`
Array	n	n	n	n	n
Sorted Array	$n \log n$	$\log n$	n	1	$\log n$
Direct Access Array	u	1	1	u	u
Hash Table	$n_{(e)}$	$1_{(e)}$	$1_{(a)(e)}$	n	n

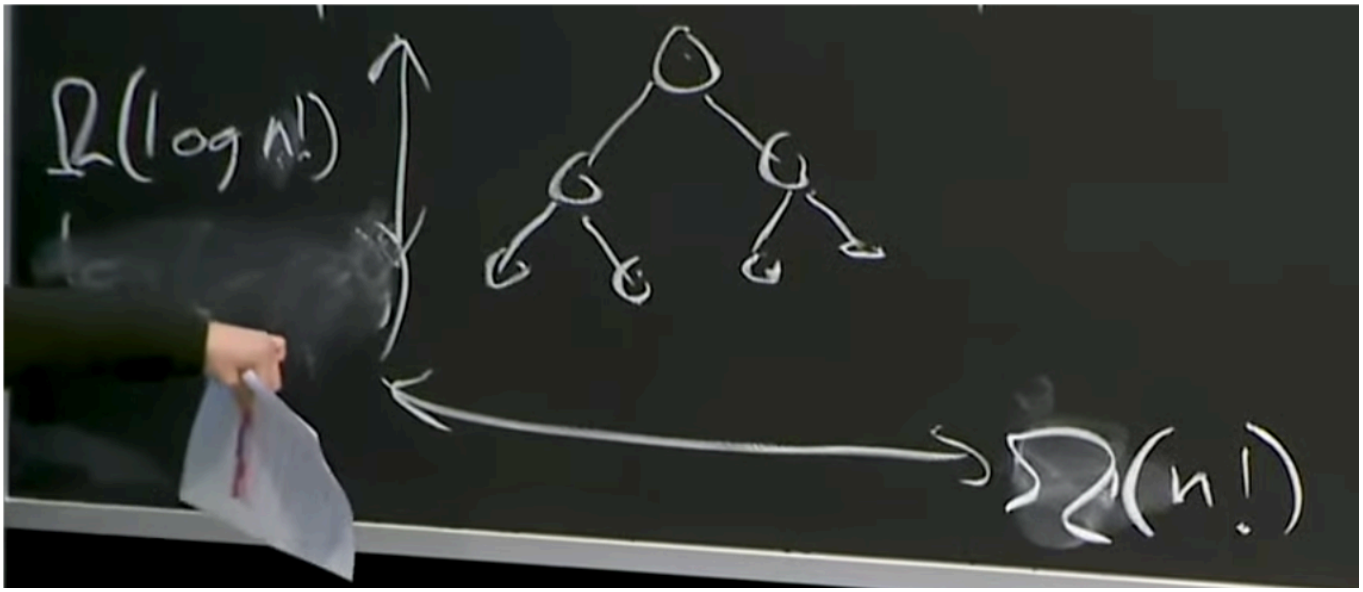
Sorting Alogrithms

today we back to sort



Comparsion Model

Sorting [ ], there are $n!$ permutations of n.



$n! = 1 * 2 * 3 \ldots * n$, take half of the sequence which bigger than n/2,
$(> n/2)! >= (n/2)^{(n/2)} = \Omega(n \log(n))$,
so the *merge sort* is the best way we can do. we can't do better in comparison model.

DAA(direct access array) Sort

u is the upper bound of array.
n is the numbers of sorted items.

Direct store by sort keys

[0 k u-1]
1.make DAA O(u)
2.Store items x in index x.key O(n)
3.walk down direct access array, and return items seen in order
4.needs: O(n + u) when u is small & keys are unique. it means can be written in one word length.

WHEN $u < n^2$

written $k => (a, b),\ a = k//n,\ b = k\%n,\ k = an + b$
`a,b = divmod(k,n)`
k is more sensitive about a than b

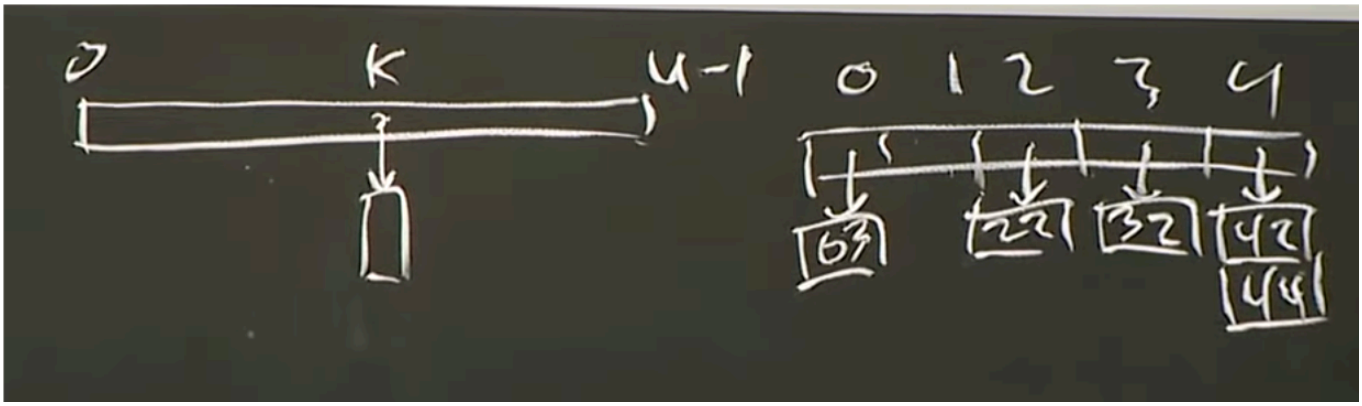
- EX:
  n = 5
  [17, 3, 24, 22, 12] to [(3,2), (0,3), (4,4), (4,2), (2,2)]
- Tuple Sort 1 by significant first
  [03 22 32 42 44] sort by a (significant thing)

[22 32 42 03 44] then sort by b
it doesn't run the way.
most significant first bad!

- Tuple Sort 2 by significant second
  [32 42 22 03 44] sort by b
  [03 22 32 *44*->42 *42*->44] sort by a stable.
  -> if 44 & 42 's orders are stable we called *stable sorting algorithm*, if it doesn't stable, then we can't take 42 be front of 44 stably.

Counting Sort

A array of space u. Sort in-place with k and a pointer to a *Sequence datastruct*.



the example of upper.
runtime: O(n + u). Beacuse *instantiate a array size of u & store O(1) for n items*.
Just sure the order of items come in order if they are collide.

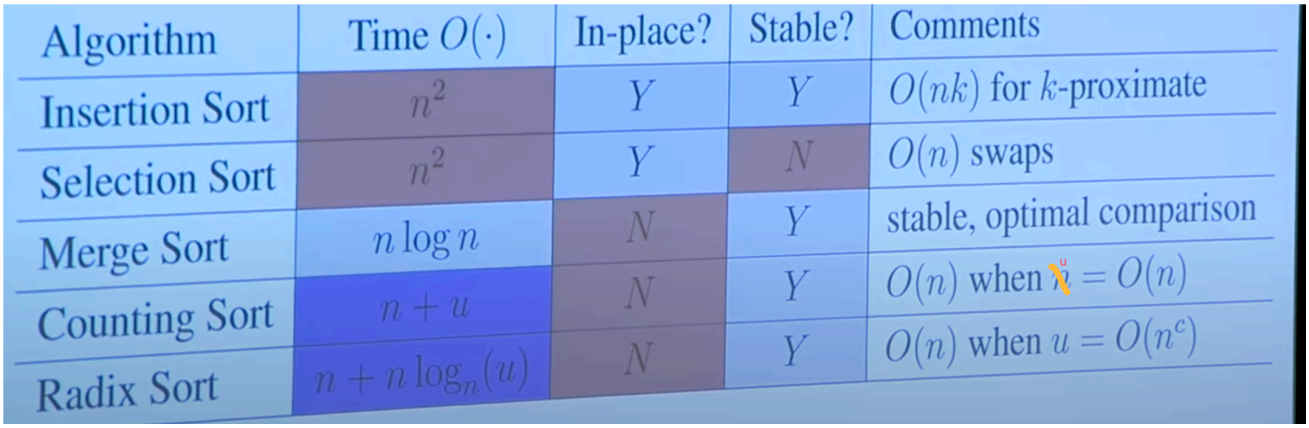
- **If [03 22 32 42 44] are 10 base, the sort is counting sort. If numbers are 5 base, the sort is the core of Radix Sort**

Radix Sort

break up integers max size u into a base and tuple. Use counting sort to store in daa

- number of digital $log_n U$.
- tuple sort on digits using counting sort from least to most significant.

- runtime: O(n *init of array* + n($\log_n U$)*sort of digital*). if $u < n^c$ ->linear time $C * n$



Algorithm	Time $O(\cdot)$	In-place?	Stable?	Comments
Insertion Sort	n^2	Y	Y	$O(nk)$ for k-proximate
Selection Sort	n^2	Y	N	$O(n)$ swaps
Merge Sort	$n \log n$	N	Y	stable, optimal comparison
Counting Sort	$n + u$	N	Y	$O(n)$ when $u = O(n)$
Radix Sort	$n + n \log_n(u)$	N	Y	$O(n)$ when $u = O(n^c)$

Comparsion model And DAA can't be compared

- Comparsion Model can take in any input, DAA need digital in a limit array.
- CM mainly care abut operation times & DAA mainly care about the cost of space and access times.s