COL 702 Lecture 9 Aug 24

Sorting without comparisons

By counting # of smaller category elements we can get the rank.

If our input is from range $1, 2, \ldots m$ then in an input of size n we can count #1's #2's .. #m from which the output can be computed uniquely

Input 4 1, 3, 1, 10, 100 . .

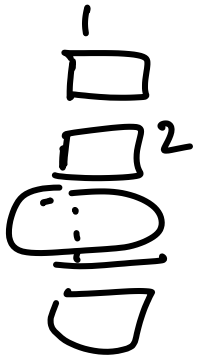
#1's	#2's	#3's	4		
3	5	1	0	–	–

Output 111 (3) 2 2 2 2 2 (5) ~ . . . . m

Count sort scans the input to find the #i's in the input and then writes the output

Running time : $O(n + m)$ ← initialize. must visit all counters to output

Assuming that hashing to the correct counter takes $O(1)$ time

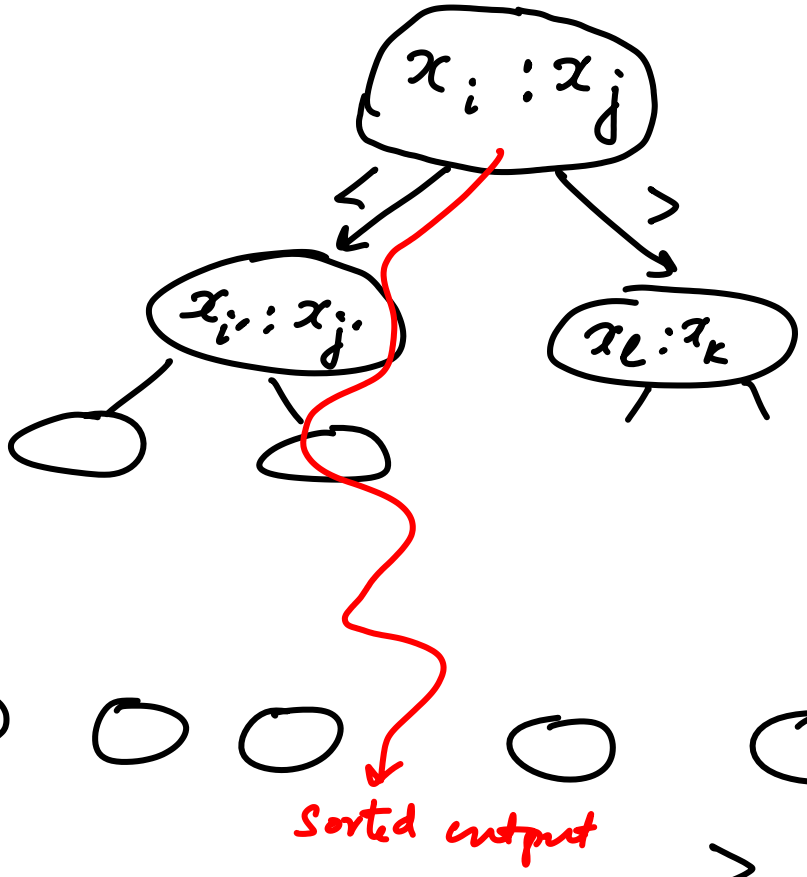


When $m \sim n$ - then 'ideal'

say $m = n$ $O(n)$

What happens if $\Omega(n \log n)$ lower bound

Comparison Tree model



Any distinct input will traverse a fixed path in the tree

$\geq n!$ leaf nodes corresponding to each perm

Obs A binary tree with L leaf nodes must have a path of length at least $\log_2 L$ $L = n!$

Since any comparison based algorithm can be represented by a $n!$ leafnode binary tree the worst case bound is $\Omega(\log(n!)) \sim \Omega(n \log n)$

Sorting strings of fixed sizes

say, each string has length l (l bits)

s_1 $\underbrace{\times \; 1 \; \times \; 0 \; \times \; 01}_{l}$

s_2

s_3

$\vdots$

s_n

$m = 2^l$

Time: $O(n + 2^l)$

$l : \leq \log n$

Radix sort partition the l bits in chunks of $\log n$ bits and repeatedly count sort on those chunks starting from the lowest significant bits

Ex:

$56, \; 21, \; 58, \; 31, \; 35, \; 29, \; 19, \; 24$

First round $21, 31, 24, 35, 56, 58, 29, 19$

2nd round $19, 21, 24, 29, 31, 35, 56, 58$

Starting with the second round, the sorting must be "stable"

Time for radix sort on

$\frac{l}{\log n}$ × Time for each round

$= O\left(n \cdot \frac{l}{\log n}\right)$

If $l = c \cdot \log n$ c : constant

time to sort is $cn = O(n)$

What about strings with variable lengths?

$l_1, l_2, l_3, \ldots l_n$

length of string $s_i = l_i$

$s_1 = 000101$ ($l_1 = 3$) $s_{10} = \underbrace{110110000}_{l_{10} = 100}$

Look at $\max_i l_i = L$

Convert all strings to length L by leading 0's

Time : $O\left(\frac{L}{\log n} \times n\right) = O\left(\frac{Ln}{\log n}\right)$

Input size of the problem was

$= \sum l_i = N$

By appending leading bits it becomes

$L \cdot n$

Compare N . Ln

Blowup can be by a quadratic factor

Goal: $O(N)$ algorithm for sorting variable size strings

Tries: Can they be used to sort?

(Digital-trees)