

CNAPS 201

S-10-10

11

• Comparison Sorts:

vs.

• non-comparison sorts:

Thm

Any Comp. sort performs $\Omega(n \log n)$

comparisons on $A[1 \dots n]$ in

a.) worst case, and

b.) average case.

Why?

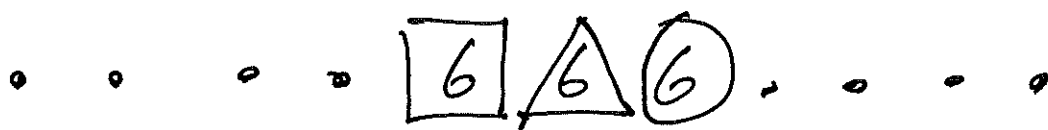
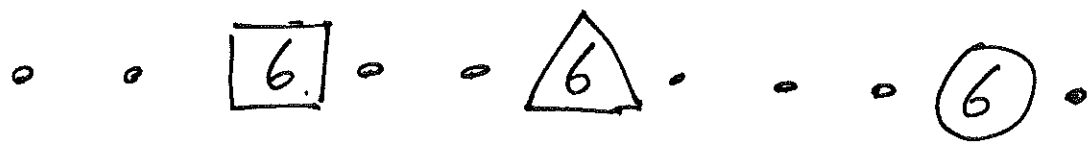
$$\lg(n!) = \Theta(n \log n).$$

Continue Counting Sort:

- if i appears in $A[i]$ then i is the $C[i]$ th order statistic.
- Thus $A[i]$ is the $C[A[i]]$ th order statistic.
- \Rightarrow If $A[i]$ contains distinct elements, then $A[i]$ would belong in position $C[A[i]]$ in $B[i]$.
- line 8 Does exactly that
- line 9 places like elements in an adjacent position in $B[i]$.

- Counting Sort is stable in the sense that elements of same value in $A[]$ are placed in $B[]$ in the 'same order' that they appear in $A[]$.

note 'same order' only make sense if there is some associated satellite data for each array element,



- Run time: Basic op. is assignment \leftarrow

$$T(n, k) = k + 1 + n + k + 2n = 3n + 2k + 1 = \Theta(n+k)$$

Radix Sort :

Assume each element of $A[]$ is a d -digit number

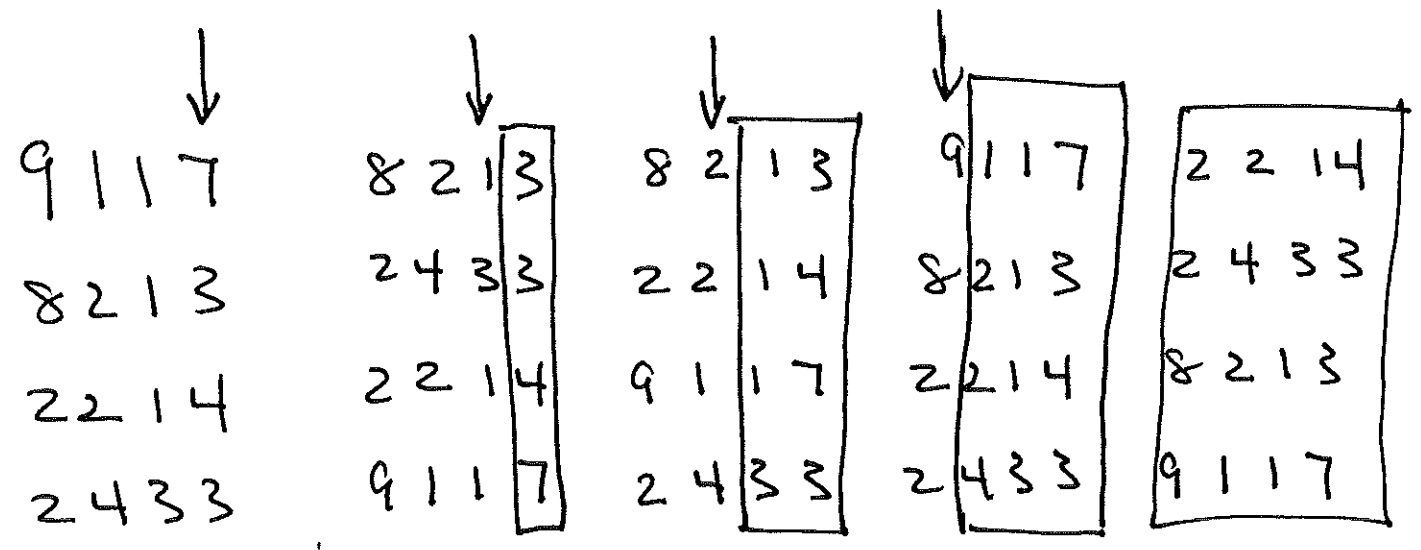
$$A[i] = x_d x_{d-1} \cdots x_2 x_1 \quad d = \# \text{ digits}$$

↑
↑
 most significant least significant

RadixSort(A, d) (Pre: \uparrow)

- 1.) for $i \leftarrow 1$ to d
- 2.) sort $A[]$ on digit i using a stable sort.

Ex. $d = 4$, $\text{length}[A] = 4$.



Run time: Assume we use counting sort on line 2. Then what is k ? If base (radix) is b , then $k = b - 1$, so

$$T(n) = \Theta(d(n + b - 1)) = \Theta(dn)$$

Bucket sort :

Assumes $A[]$ consists of n #s in $[0, 1)$, uniformly distributed.

Divide $[0, 1)$ into n sub-intervals of equal length $= \frac{1}{n}$

$$[0, \frac{1}{n}), [\frac{1}{n}, \frac{2}{n}), [\frac{2}{n}, \frac{3}{n}), \dots, [\frac{n-1}{n}, 1)$$

note: expected # of array elements in each sub-interval is 1.

uses an array $B[1 \dots n]$ of linked lists (i.e. Buckets), i.e.

$B[i]$ is a linked list of Real #s in range 0 to 1 $\in [0, 1)$.

BucketSort(A)



1.) for $i \leftarrow 1$ to n

2.) insert $A[i]$ into $B[\lfloor nA[i] \rfloor + 1]$

3.) for $i \leftarrow 1$ to n

4.) sort list $B[i]$ using insertion sort

5.) concatenate lists $B[1], B[2], \dots, B[n]$

6.) return new list.

note : • input $A[1 \dots n]$

• output is a list.

why does x belong in the
list (Bucket) $B[\lfloor nx \rfloor + 1]$

8

observe that for $x \mapsto nx$ maps:

$$[0, 1) \longrightarrow [0, n)$$

hence maps sub-intervals:

$$[0, \frac{1}{n}) \longrightarrow [0, 1)$$

$$[\frac{1}{n}, \frac{2}{n}) \longrightarrow [1, 2)$$

$$[\frac{2}{n}, \frac{3}{n}) \longrightarrow [2, 3)$$

⋮

$$[\frac{n-1}{n}, n) \longrightarrow [n-1, n)$$

thus $x \in [\frac{j-1}{n}, \frac{j}{n})$ is mapped to

$$nx \in [j-1, j). \quad \therefore \lfloor nx \rfloor = j-1$$

$$\therefore \lfloor nx \rfloor + 1 = j$$

Run time:
 note the ^{total} cost of all ops, other than line 4 is $\Theta(n)$.

If there are n_j elements in $B[j]$, then cost of line (4) is $\Theta(n_j^2)$.

So avg. cost of Bucket sort is

$$T(n) = \Theta(n) + \sum_{j=1}^n \text{const} \cdot n_j^2$$

Recall: expected value of n_j is 1.

$$\therefore T(n) = \Theta(n)$$

Lower Bounds & Computational Complexity

◦ Consider the set of all algorithms that solve some problem P

◦ we have two goals

(1) find an algorithm that solves P in (worst case say) time $O(f(n))$ for some $f(n)$, that we wish to reduce, as far as possible asymptotically

(2) Prove that any algorithm that solves P must run in (worst case) time $\Omega(g(n))$, for some $g(n)$ that we seek to increase as much as possible.

[11]

we're happy when $f(n) = \Theta(g(n))$,
for then we know we have
a best possible algorithm

(1) is called Algorithmics

(2) is called Complexity theory,

two techniques for (2):

- Decision tree arguments
(information theoretic lower bounds)
- Adversary arguments

See Brassard & Rotelety