CMPS 201 5-10-10

• 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 $A[\,]$ then i is the $C[i]^{th}$ order statistic.
- thus $A[j]$ is the $C[A[j]]^{th}$ order statistic.
- If $A[\,]$ contains distinct elements, then $A[j]$ would belong in Position $C[A[j]]$ in $B[\,]$.
- line 8 Does exactly that
- line 9 Places like elements in an adjacent Position in $B[\,]$.

- Counting Sort is <u>stable</u> 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 satillite data for each array element.

  . . $\boxed{6}$ . . 6 . . . 6 .

  . . . . $\boxed{6}$ 6 6 . . . .

- Run time: Basic op. is assignment ←

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

Radix sort:

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

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

most significant   least significant

RadixSort(A, d) (pre:)

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

2.) sort A[ ] on digit i using a stable sort.

Ex. $d = 4$, $length[A] = 4$.

9117	8213	8213	9117	2214
8213	2433	2214	8213	2433
2214	2214	9117	2214	8213
2433	9117	2433	2433	9117

Run time: Assum 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 consists of #s in [0, 1), uniformly distributed.

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

$$[0, \tfrac{1}{n}), [\tfrac{1}{n}, \tfrac{2}{n}), [\tfrac{2}{n}, \tfrac{3}{n}), \ldots, [\tfrac{n-1}{n}, 1)$$

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

uses an array B[1 ... n] of linked lists (i.e. Buckets), i.e. B[i] is a linked list of Real #s in range 0 to 1 $\doteq$ [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 $B[1], B[2], \ldots, B[n]$

6.) return new list.

note: • input $A[1 \ldots n]$

• output is a list.

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

observe that fcn $x \mapsto nx$ maps:

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

hence maps sub-intervals:

$$\left[0,\tfrac{1}{n}\right) \longrightarrow [0,1)$$

$$\left[\tfrac{1}{n},\tfrac{2}{n}\right) \longrightarrow [1,2)$$

$$\left[\tfrac{2}{n},\tfrac{3}{n}\right) \longrightarrow [2,3)$$

$$\vdots$$

$$\left[\tfrac{n-1}{n},n\right) \longrightarrow [n-1,n)$$

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

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

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

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 Bucketsort is

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

<u>Recall</u>: 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, asymptotically as far as possible

(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.

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

(1) is called <u>Algorithmics</u>
(2) is called <u>Complexity theory</u>,

two techniques for (2):

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

See Brassard & Bratley.