DSA [02/28/2024]

(BASICALLY)

COUNTING SORT
RADIX SORT
BUCKET SORT) — O(n) → REASONABLE ASSUMPTIONS & USE CASE

→ ONLY LOOKS AT 1 ITEM (ALT. BENCHMARK)

A **DECISION TREE** THAT SORTS "n" ELEMENTS HAS HEIGHT AT LEAST nlogn

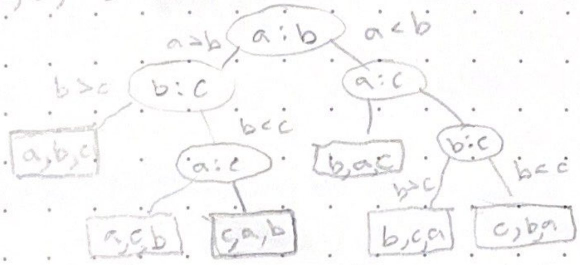
n ITEMS — HOW MANY WAYS TO SORT THEM? ⟶ n!

3 ITEMS — a, b, c :

abc
acb
bac
bca
cab
cba

} 3!

SORT [a, b, c] DECISION TREE



HEIGHT $h = 2^h$ LEAVES

$h \geq \lg(n!)$

$h \geq n \lg n$

(THEORETICAL BEST)

THIS IS WHY SORTING ROUTINES ARE $O(n \lg n)$

DSA [02/28/2024]

COUNTING SORT

→ ALL n ELEMENTS ARE IN RANGE 0...k IF k≈n, k<n, k<<n

```
                       INPUT     RESULT
                         ↓         ↓
1  COUNTINGSORT (array A, array B, range k){  ←  O(n+n+k)
2     new array C[k+1] = {0};             } INIT COUNTING ARRAY
3     for(i=0; i < A.Length; i++)         } COUNT FREQUENCY
4        C[A[i]]++;                         OF VALUES
5     For(i=1; i<=k, i++)                 } ACCUMULATE COUNT
6        C[i] += C[i-1];                    TRANSLATE TO POSITION
7     For (i= A.length-1; i >=0; i--) {   } PLACE VALUE'S into
8        C[A[i]]--;                         PROPPER PLACE BY
9        B[C[A[i]]] = A[i];                 (RANGE OF VALUES)
10    }                                     PREVIOUS CALCULATION
11 }
```

LINE

2: INIT COUNTING ARRAY

3-4: COUNT # OF TIMES ELEMENT APPEARS

5-6: ACCUMULATE THE COUNTS → TRANSLATES COUNTS INTO POSITIONS.

7-10: PLACE VALUES INTO PROPER PLACE BY (RANGE OF VAL) PREV. CALC

$A[0\ \overset{A}{2}\ \overset{A}{3}\ 1\ \overset{B}{3}\ \overset{B}{2}\ \overset{C}{2}\ \overset{C}{3}]$

$i=7...0$

$B[\underset{0}{0}\ \underset{1}{1}\ \underset{2}{\overset{A}{2}}\ \underset{3}{\overset{B}{2}}\ \underset{4}{\overset{C}{2}}\ \underset{5}{\overset{A}{3}}\ \underset{6}{\overset{B}{3}}\ \underset{7}{\overset{C}{3}}]$

11,1 0

$C[\underset{0}{1}\ \underset{1}{1}\ \underset{2}{3}\ \underset{3}{3}]$ "3 3's"

(AFTER) $C[0\ 1\ 2\ 5]$

STABLE SORT

- PRESERVES ORDER IN CASE OF A TIE

(BY COMMON IMPLEMENTATION)

STABLE	NOT STABLE
BUBBLE	HEAP SORT
INSERTION	QUICKSORT
BUCKET/RADIX	
COUNTING	
MERGE	

DSA [02/28/2024]

RADIX SORT

LEAST TO MOST SIG! MUST USE STABLE SORT!

DIGITS IN NUM

LEAST - SIGNIFIGANT TO MOST

```
RADIXSORT (A, d) {
    For (i = 1 ; i<=d ; i++)
        USE AN O(n) STABLE SORT TO SET ARRAY A ELEMENTS
        BY ITS i^th DIGIT
}
```

BUCKET SORT

[ ] LENGTH

```
BUCKET-SORT (A, n)
    For(i=0; i<n; i++)
        INSERT A[i] INTO BUCKET B[A[i]]       } NEED MULTIPLE BUCKETS TO KEEP EACH SMALL IN SIZE
    FOR(i=0; i<n; i++)
        SORT BUCKET B[i] WITH INSERTION SORT  } INSERTION SORT   O(n^2)
```

RADIX-BUCKET SORT (HYBRID)

LARGEST VALUE (INCLUDING 0)

d - ALWAYS 9

K - RANGE OF EACH DIGIT 7/0

0-9

"ARRAY OF VECTORS"

[ ] SIZE # OF DIGITS

```
RADIX-BUCKET SORT (A, n, d) {
    "LIST" BUCKET[10]
O(d)    FOR (i=1; i<=d; i++){   <- i^th DIGIT FROM LSD TO MSD (UNITS, TENS, HUND.
O(n)        FOR (j=0; j<n; j++) {
                r = i^th = digit OF A[j];
                BUCKET[r]. PUSH_BACK (A[i]);
            }
O(k)        FOR( j=0, j<10, j++) {
                ADD ITEMS FROM BUCKET[j] INTO A IN ORDER
            }
    }
}
```

10<u>3</u>

K IN OUR CASE 0-9

DSA

RADIX BUCKET SORT EXAMPLE

$d=3$ $k=3$

A	103	202	222	102	101	213

(1) SORT BY 1's PLA

INTO BUCKETS

	101	102 222 202	213 103	
0	1	2	3	

INTO A[ ]

A[101, 202, 222, 102, 103, 213]

INTO BUCKET

(2) SORT BY 10's DIGIT

103 102 202 101	213	222	
0	1	2	3

A[101, 202, 102, 103, 213, 222]

	103 102 101	222 213 202	
0	1	2	3

A[101, 102, 103, 202, 213, 222] ← COMPLETE!

5000, 3000, 2000, 1000 ← ENTRIES

RUN RAW AGAINST FILES → GET TIMES

RUN WITH CONSTANT → GET TIMES

CONSTANT SHOULD BE < OR = RAW?

YOU DONT NEED ENTRIES PER SECOND

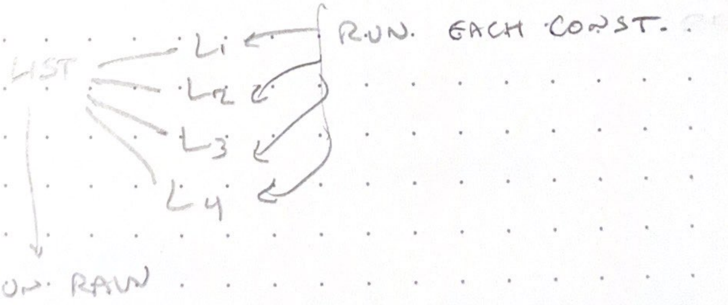
COMPARE CONSTANTS!

CONST	SIZE	TIME	SIZE/TIME
L1 1000	1K	1?	
L2 2000	5K	?	
L3 3000	10R	?	
L4 4000	100R	?	
L4 5000	1M	?	

↑ ENTRIES/SEC

TEST AGAINST NO CONST (RAW)

(RAW) SIZE/TIME VS. SIZE/TIME (WITH CONSTANT)


LIST
L1
L2
L3
L4
RUN EACH CONST.
RUN RAW