2021.03.15 Chi-Wen Teng 鄧祺文

CLOSURE REPORT FOR ALGORITHM LAB WEEK 2： NON-COMPARISON SORT

QUESTION 1:

Function "sort_by_digit" is stable counting sort. If we rewrite the function to an unstable version, can radix sort still work correctly?

The radix sort can still work correctly because we only care about the "order of the value", not the "order of the exact element". For a stable sorting algorithm, it means that with two same value elements, the former one will always stay in front of the latter. However, for a unstable sorting algorithm, it means that with two same value elements, the former one's position might exchange with the latter one's position. In usual case, these two types are muchly the same, yet if you use for something like leaderboard ranking, you can only pick stable sorting for ranking stabilization, or you will find that as soon as you refresh the page, the ranking for the same score might keep changing.

QUESTION 2:

This radix implement is processing through least significant digit to most significant digit. Can you design radix algorithm which start at most significant digit?

Most significant digit radix sort:

1. Take the most significant digit of each key
2. Sort the list of elements based on that digit, grouping elements with the same digit into one bucket

3. Recursively sort each bucket, starting with the next digit to the right
4. Concatenate the buckets together in order

Example: Sort the list of 170, 045, 075, 090, 002, 024, 802, 066

1. Sorting by MSD (100s place) gives:

Zero hundreds bucket: 045, 075, 090, 002, 024, 066

One hundreds bucket: 170

Eight hundreds bucket: 802

2. Sorting by next digit (10s place) is only needed for zero hundreds bucket because no other hundreds buckets has more than one elements:

Zero tens bucket: 002

Twenties bucket: 024

Forties bucket: 045

Sixties bucket: 066

Seventies bucket: 075

Nineties bucket: 090

3. Sorting by least significant digit (1s place) is not needed because there is not other tens bucket has more than one elements. Thus, we concatenate all of the bucket in order will get:

002, 024, 045, 066, 075, 090, 170, 802

The sorting progress has finished.

QUESTION 3:

Please design an algorithm to measure minimum usable 'd' of list 'A' and radix r.

For radix 'r':

1. Find the biggest value element in the list
2. Divide all of the elements in the list from 2 to the square root of the biggest value element in the list

3. Count the value type of the remainder and find the smallest amount type
4. From the smallest amount type, mark down the divisor it use and that will be the best radix 'r' in the list (using the least amount of bucket in the list)

For 'd' of list 'A':

1. Divide the elements in the list with the perfect 'r' we find in the above method
2. Compare the quotient results from the calculations, find the largest number in it (if it has a remainder, need to add one)
3. The result will be the smallest possible usable 'd' of list 'A'

QUESTION 4:

Please analysis the space complexity and the time complexity. (Should be funcitons with arguments n, r, d)

Time Complexity:

Best Case: $\Omega(d * (n + r))$

Worst Case: $O(d * (n + r))$

Average Case: $\Theta(d * (n + r))$

Distribute data into bucket needs O(n)

Concatenate data from the bucket needs O(r)

>> Every round needs O(n + r)

'd' rounds for the total progress

>> Total rounds need O(d * (n + r))

Space Complexity:

'r' buckets with 'n' capacity for elements

>> O(n * r)