Harsimran Singh Dhillon. B number - B00983136

Design and Analysis of Algorithms.

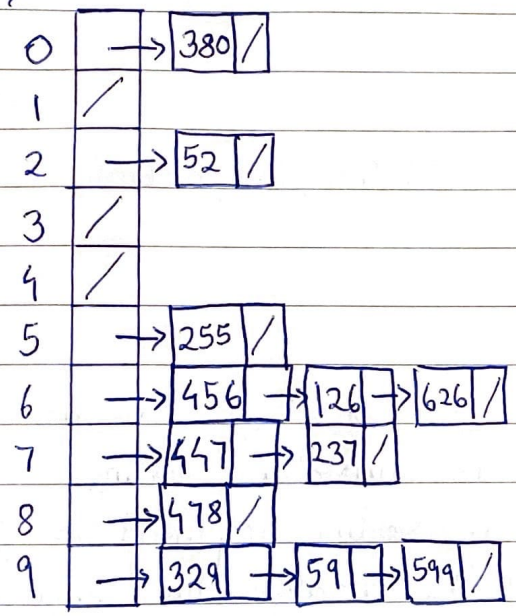
Theory Assignment - 2.3

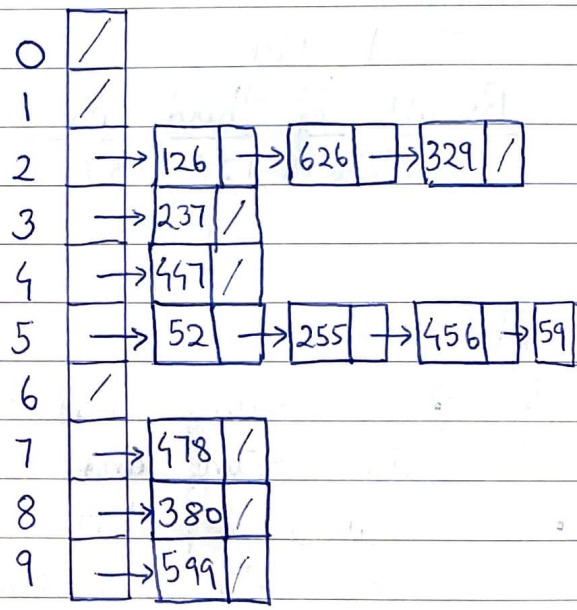
1] Use the radix sort algorithm to sort the following numbers. Treat every data as a 3-digit integer.

456, 329, 478, 59, 52, 447, 380, 126, 237, 599, 626, 255

a] Draw a figure to show the actions step by step by treating each digit as a "digit".

→





First Pass

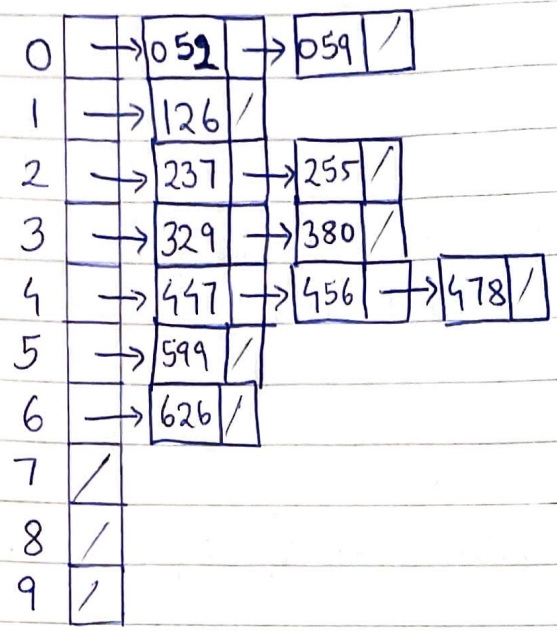
Second Pass

Result of First Pass :-

380, 52, 255, 456, 126, 626, 447, 237, 478, 329, 59, 599

Result of Second Pass:-

126, 626, 329, 237, 447, 52, 255, 456, 59, 478, 380, 599



Third Pass

Result of third pass:-

052, 059, 126, 237, 255, 329, 380, 447, 456, 478, 599, 626

b]

Ans:-) In sorting algorithms, there are two types one is stable sorting and another one is unstable sorting.

- It is very important to have stable sorting implementation for certain sorting algorithms to fulfil the algorithms requirement to provide sorted output.
- Consider a scenario in radix algorithm, if we have elements like [illegible], 245, 345, 735, 835
- So for the first iteration / Pass of radix algorithm we will get [illegible], 345, 245, 735, 835

Here if you see that the element 5 is present in all the elements. But the relative position of the elements got changed. i.e 245 and 345 changed their position in wrong manner. This is because of unstable sorting.

Furthermore if we perform 2nd pass we will get;

835, 735, 345, 245

- Again here the relative position has been changed for 345 and 245 as well as 835 and 735.
- So for in between each pass and step, because of unstable sorting the relative position gets changed. Resulting in the change of the order of the elements being sorted.
- This will impact our output. Where as, this wont be the case scenario in stable sorting.
- If stable sorting is used then we would get the elements :- 245, 345, 735, 835 as our output after the first pass. and 735, 835, 245, 345 as our output after the second pass. This maintains the relative position of the element and provides stability to our result.
- As we are sorting based on our second element of a number for eg (735, 835) in this we are sorting based on 3 and both the numbers are equal based on our 2nd digit. This stable sorting algorithm maintains the relative position i.e it follows the stability property.

c] Describe what conditions should be met for radix sort to be O(n)?

→ We know that the time complexity of radix sort is O(d x n)

Here d is a constant and represents the number of digits. and n is the number of ~~digits~~ elements.

To achieve the time complexity as O(n), following scenarios should be considered.

1] Smaller range of values :-

We would be able to make radix sort efficient if we are working on a smaller range of values as compared to the number of elements. If the

range is not smaller, then there would be impact on storage and time complexity. As higher range values will make it difficult to store values - by adding and to traverse these elements will increase the time complexity.

3] Uniform distribution of elements :-

As we work on digits starting from 0 to 9. So it would make it easier and efficient if we are able to get elements distributed evenly in this spectrum from 0 to 9. This will make the traversal as well as storing the elements efficient. If they are not uniformly distributed and densely populated in specific region, then this will decrease the efficiency in both the ways storage as well as traversal.

3] Fixed length of elements :-

If the length is fixed and known to us then we will be able to get the number of passes required to achieve the required output. For eg if we have 3 digit elements in our input set. then we would require 3 passes. If this length is fixed we will be able to get O(n) as our time complexity. If the length is variable then the time complexity will depend on O(dxn) where d being variable will impact the time complexity.

Q2] Suppose we want to apply Radix sort to sort 100,000 4 letter words with each letter taken from the English alphabet (26 letters, all lower cased). Assuming that the running time for sorting n elements within range 1...k using Counting sort is 2n+2k, calculate the running time for each of the following strategies.

a] Treat letters at each of the four positions as a digit.

Given:- n = 100,000

k = 26.

Formula :- 2n+2k

For pass 1 :- 2n + 2k

= 2x100000 + 2x26

= 200,052.

This is the running time for 1st pass.

Similarly we have 3 more passes, because we have total 4 digits.

So our total running time will be

= 4 (Time required for pass 1)

= 4 (200,052)

= 800,208

This is the total running time which is required combining the 4 passes required to sort the elements, as we have 4 positions as a digit.

b] Treat 2-letters subwords at position 1-2 as a digit and 2-letter subwords at position 3-4 as another digit.

Given:- n = 100,000

k = 26

As we will be considering 1-2 as a single digit and position 3 and 4 as a another single digit.

So now total we have 2 digits to be compared.

So the number of passes required will be 2 to sort the elements.

Here we will have modification k will become $k^2 = (26)^2$

For Pass 1:- $2n + 2k^2$

$= 2 \times 100,000 + 2 \times 26^2$ (both letters have range of 26)

$= 200000 + 1352$

$= 201352$

the Value of k became k^2 over here because we had 2 letters being considered as 1 digit.

Similarly we will calculate for Pass 2.

For Pass 2:- $2n + 2k^2$

$= 2 \times 100,000 + 2 \times 26^2$

$= 200000 + 1352$

$= 201352$.

So, the total running time will be = Running time for Pass 1 + Running time for pass 2

$= 201352 + 201352$

This is our total running time = 402,704.

c] Treat all 4 letters as a digit

Given:- $n = 100,000$

$k = 26$

Since here we are told to consider all the 4 letters as a single digit.

for 1 letter $k = 26$

So, for 4 letter $k = 26 \times 26 \times 26 \times 26$

all letters will have the range of 26.

So our modified k will become $k^4 = (26)^4$

Since We have only one digit. So we would require only 1 pass to sort.

For pass 1 :- $2n+2k$

$= 2 \times 100,000 + 2 \times (26)^4$

$= 2,00,000 + 2 \times 456976$

$= 200000 + 913952$

$= 1,113,952$

This will be our running time.

d] Which strategy is the best strategy to minimize the running time?

Sol For strategy a] where each letter was considered as a digit we get 800,208.

For strategy b] where ~~each~~ letter two letter were considered as a digit we get 402704.

For strategy c] where 4 letters were considered as a digit we get 1,113,952.

So from the above scenarios, strategy b] is the best strategy to minimize as we get the least running time complexity. As we get to learn from this, that combining letters ~~wd~~ as to form a single digit will reduce time complexity. But this technique only works till certain combination and would not prove to be efficient if we combine all the letters to form a single digit.