Sorting Algorithms

Content

Radix Sort

About

The idea of Radix Sort is to do digit by digit sort starting from least significant digit to most significant digit. Radix sort uses counting sort as a subroutine to sort.

- Original, unsorted list:
- 170, 45, 75, 90, 802, 24, 2, 66
-
- Sorting by least significant digit (1s place) gives:
- [*Notice that we keep 802 before 2, because 802 occurred
- before 2 in the original list, and similarly for pairs
- 170 & 90 and 45 & 75.]
-
- 170, 90, 802, 2, 24, 45, 75, 66
-
- Sorting by next digit (10s place) gives:
- [*Notice that 802 again comes before 2 as 802 comes before
- 2 in the previous list.]
-
- 802, 2, 24, 45, 66, 170, 75, 90
-
- Sorting by the most significant digit (100s place) gives:

i. 2, 24, 45, 66, 75, 90, 170, 802

Implementation

https://github.com/lavishabhambri/Weekly-Algo-Newsletter/blob/main/Sorting/Codes/RadixSort.cpp

What is the running time of Radix Sort?

Let there be d digits in input integers. Radix Sort takes O(d*(n+b)) time where b is the base for representing numbers, for example, for the decimal system, b is 10. What is the value of d? If k is the maximum possible value, then d would be O(logb(k)). So overall time complexity is O((n+b) * logb(k)). Which looks more than the time complexity of comparison-based sorting algorithms for a large k. Let us first limit k. Let k <= nc where c is a constant. In that case, the complexity becomes O(nLogb(n)). But it still doesn't beat comparison-based sorting algorithms.
What if we make the value of b larger?. What should be the value of b to make the time complexity linear? If we set b as n, we get the time complexity as O(n). In other words, we can sort an array of integers with a range from 1 to nc if the numbers are represented in base n (or every digit takes log2(n) bits).

Applications of Radix Sort

1. In a typical computer, which is a sequential random-access machine, where the records are keyed by multiple fields, radix sort is used. For eg., you want to sort on three keys month, day and year. You could compare two records on year, then on a tie on month and finally on the date. Alternatively, sorting the data three times using Radix sort first on the date, then on month, and finally on year could be used.
2. It was used in card sorting machines that had 80 columns, and in each column, the machine could punch a hole only in 12 places. The sorter was then programmed to sort the cards, depending upon which place the card had been punched. This was then used by the operator to collect the cards which had the 1st row punched, followed by the 2nd row, and so on.

Counting Sort

About

Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (kind of hashing). Then doing some arithmetic to calculate the position of each object in the output sequence.

Example

For simplicity, consider the data in the range 0 to 9.

Input data: 1, 4, 1, 2, 7, 5, 2

1. Take a count array to store the count of each unique object.

Index:	0	1	2	3	4	5	6	7	8	9
Count:	0	2	2	0	1	1	0	1	0	0

2. Modify the count array such that each element at each index stores the sum of previous counts.

Index:	0	1	2	3	4	5	6	7	8	9
Count:	0	2	4	4	5	6	6	7	7	7

   The modified count array indicates the position of each object in the output sequence.

3. Rotate the array clockwise for one time.

Index:	0	1	2	3	4	5	6	7	8	9
Count:	0	0	2	4	4	5	6	6	7	7

4. Output each object from the input sequence followed by increasing its count by 1.

   Process the input data: 1, 4, 1, 2, 7, 5, 2. Position of 1 is 0.

   Put data 1 at index 0 in output. Increase count by 1 to place next data 1 at an index 1 greater than this index.

Points to be noted

1. Counting sort is efficient if the range of input data is not significantly greater than the number of objects to be sorted. Consider the situation where the input sequence is between range 1 to 10K and the data is 10, 5, 10K, 5K.
2. It is not a comparison-based sorting. Its running time complexity is O(n) with space proportional to the range of data.
3. It is often used as a subroutine to another sorting algorithm like radix sort.
4. Counting sort uses partial hashing to count the occurrence of the data object in O(1).
5. Counting sort can be extended to work for negative inputs also.

Implementation

https://github.com/lavishabhambri/Weekly-Algo-Newsletter/blob/main/Sorting/Codes/CountingSort.cpp

Time Complexity - O(n+k) where n is the number of elements in the input array and k is the range of input.

Auxiliary Space - O(n+k)