Radix Sort

The lower bound for Comparison based sorting algorithm (Merge Sort, Heap Sort, Quick-Sort .. etc) is Ω(nLogn), i.e., they cannot do better than nLogn.

Counting sort is a linear time sorting algorithm that sort in O(n+k) time when elements are in range from 1 to k.

What if the elements are in range from 1 to n^2?

We can't use counting sort because counting sort will take $O(n^2)$ which is worse than comparison based sorting algorithms. Can we sort such an array in linear time? Radix Sort is the answer. 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.

The Radix Sort Algorithm

1) Do following for each digit i where i varies from least significant digit to the most significant digit.

……**a)** Sort input array using counting sort (or any stable sort) according to the i'th digit.

Example:

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 most significant digit (100s place) gives:

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

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 decimal system, b is 10. What is the value of d? If k is the maximum possible value, then d would be $O(\log_b(k))$. So overall time complexity is $O((n+b) * \log_b(k))$. Which looks more than the time complexity of comparison based sorting algorithms for a large k. Let us first limit k. Let $k <= n^c$ where c is a constant. In that case, the complexity becomes $O(n\log_b(n))$. But it still doesn't beat comparison based sorting algorithms.

What if we make 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 range from 1 to n^c if the numbers are represented in base n (or every digit takes $\log_2(n)$ bits).

Is Radix Sort preferable to Comparison based sorting algorithms like Quick-Sort?

If we have $\log_2 n$ bits for every digit, the running time of Radix appears to be better than Quick Sort for a wide range of input numbers. The constant factors hidden in asymptotic notation are higher for Radix Sort and Quick-Sort uses hardware caches more effectively. Also, Radix sort uses counting sort as a subroutine and counting sort takes extra space to sort numbers.

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

Implementation of Radix Sort

Following is a simple implementation of Radix Sort. For simplicity, the value of d is assumed to be 10. We recommend you to see Counting Sort for details of countSort() function in below code.

Most popular in Sorting

Find the number of elements greater than k in a sorted array

Find minimum changes required in an array for it to contain k distinct elements

Sort an array of strings based on the frequency of good words in them

IntroSort or Introspective sort

Remove elements to make array sorted

Unbounded Fractional Knapsack

Divide array into two parts with equal sum according to the given constraints

Sort an array of strings according to string lengths using Map

C/C++ | Java | Python | C# | PHP

```java
// Radix sort Java implementation
import java.io.*;
import java.util.*;

class Radix {

    // A utility function to get maximum value in arr[]
    static int getMax(int arr[], int n)
    {
        int mx = arr[0];
        for (int i = 1; i < n; i++)
            if (arr[i] > mx)
                mx = arr[i];
        return mx;
    }

    // A function to do counting sort of arr[] according to
    // the digit represented by exp.
    static void countSort(int arr[], int n, int exp)
    {
        int output[] = new int[n]; // output array
        int i;
        int count[] = new int[10];
        Arrays.fill(count,0);

        // Store count of occurrences in count[]
        for (i = 0; i < n; i++)
            count[ (arr[i]/exp)%10 ]++;

        // Change count[i] so that count[i] now contains
        // actual position of this digit in output[]
        for (i = 1; i < 10; i++)
            count[i] += count[i - 1];

        // Build the output array
        for (i = n - 1; i >= 0; i--)
        {
            output[count[ (arr[i]/exp)%10 ] - 1] = arr[i];
            count[ (arr[i]/exp)%10 ]--;
        }

        // Copy the output array to arr[], so that arr[] now
        // contains sorted numbers according to curent digit
        for (i = 0; i < n; i++)
            arr[i] = output[i];
    }

    // The main function to that sorts arr[] of size n using
    // Radix Sort
    static void radixsort(int arr[], int n)
    {
        // Find the maximum number to know number of digits
        int m = getMax(arr, n);

        // Do counting sort for every digit. Note that instead
        // of passing digit number, exp is passed. exp is 10^i
        // where i is current digit number
        for (int exp = 1; m/exp > 0; exp *= 10)
            countSort(arr, n, exp);
    }

    // A utility function to print an array
    static void print(int arr[], int n)
    {
        for (int i=0; i<n; i++)
            System.out.print(arr[i]+" ");
    }


    /*Driver function to check for above function*/
    public static void main (String[] args)
    {
        int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};
        int n = arr.length;
        radixsort(arr, n);
        print(arr, n);
    }
}
/* This code is contributed by Devesh Agrawal */
```

Output:

```
2 24 45 66 75 90 170 802
```

More related articles in Sorting

Alternate XOR operations on sorted array

Maximal Disjoint Intervals

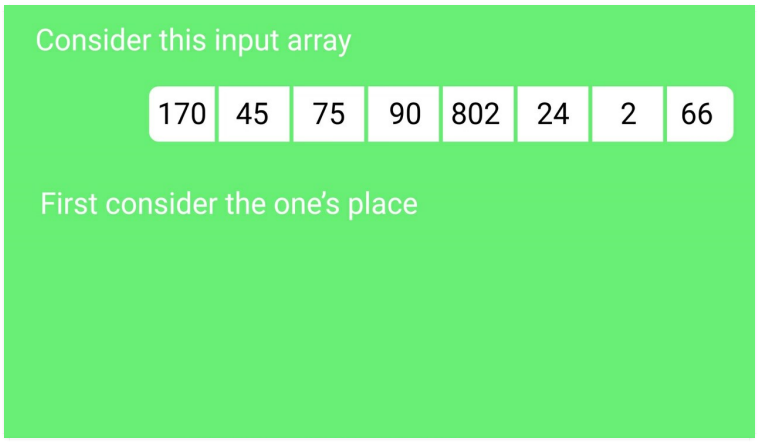
Sort ugly numbers in an array at their relative positions

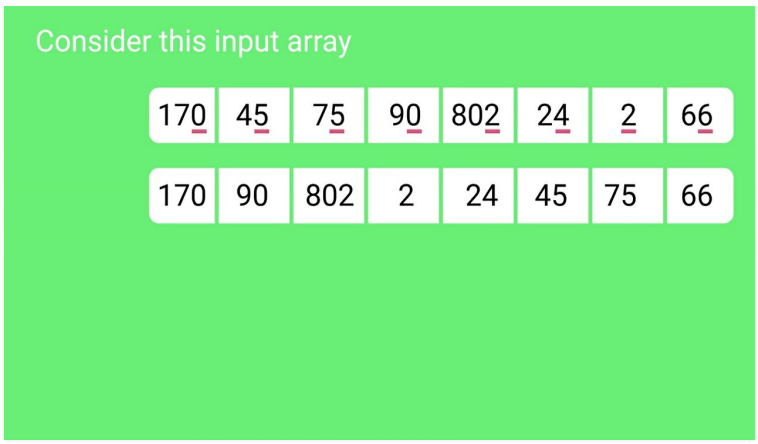
Maximum number of elements without overlapping in a Line

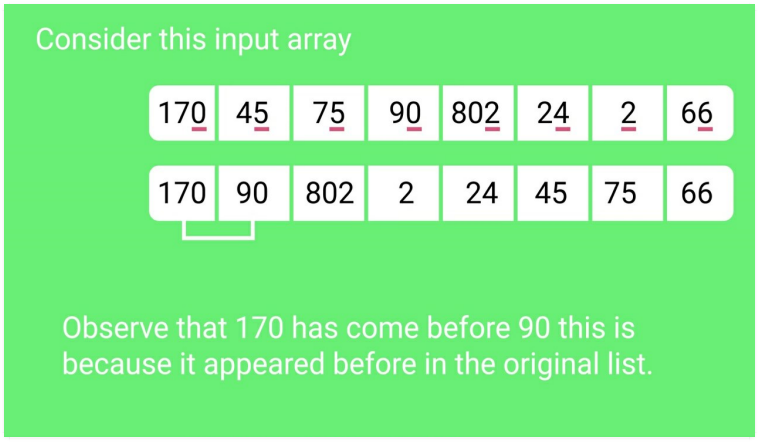
Find the minimum number of rectangles left after inserting one into another


Radix Sort | GeeksforGeeks
Watch later
Share
Consider this input array
170 45 75 90 802 24 2 66
170 90 802 2 24 45 75 66
802 2 24 45 66 170 75 90
2 24 45 66 75 90 170 802

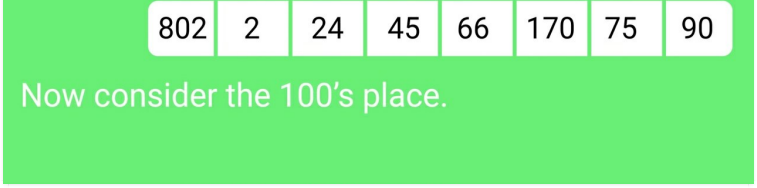
Snapshots:

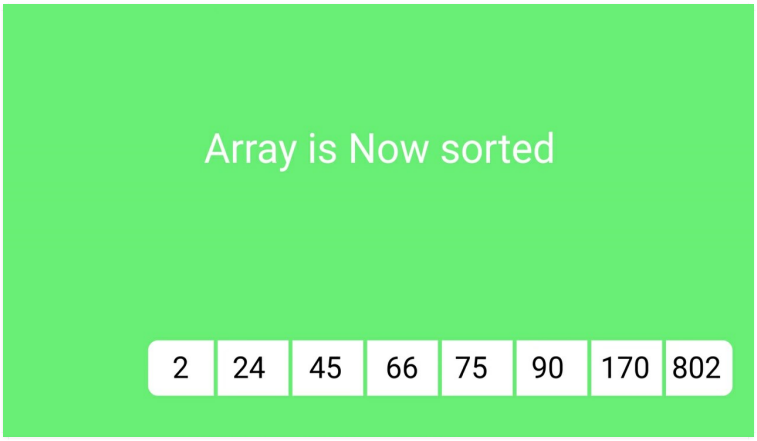

Consider this input array
170 45 75 90 802 24 2 66
First consider the one's place


Consider this input array
170 45 75 90 802 24 2 66
170 90 802 2 24 45 75 66


Consider this input array
170 45 75 90 802 24 2 66
170 90 802 2 24 45 75 66
Observe that 170 has come before 90 this is because it appeared before in the original list.


Consider this input array
170 45 75 90 802 24 2 66
170 90 802 2 24 45 75 66





Quiz on Radix Sort

Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz:

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Heap Sort
- QuickSort
- Counting Sort
- Bucket Sort
- ShellSort

References:

http://en.wikipedia.org/wiki/Radix_sort
http://alg12.wikischolars.columbia.edu/file/view/RADIX.pdf
MIT Video Lecture
Introduction to Algorithms 3rd Edition by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Sort elements by frequency | Set 1
Count Inversions in an array | Set 1 (Using Merge Sort)
Merge Sort for Linked Lists
Sort an array of 0s, 1s and 2s
std::sort() in C++ STL
Sort a nearly sorted (or K sorted) array
Sort numbers stored on different machines
Iterative Quick Sort
Sort a linked list of 0s, 1s and 2s
Counting Sort
Sort elements by frequency | Set 2
Sort n numbers in range from 0 to n^2 - 1 in linear time
Bucket Sort
Sort an array according to the order defined by another array
Time complexity of insertion sort when there are O(n) inversions?

Improved By : DrRoot_, rathbhupendra

Article Tags : Sorting

Practice Tags : Sorting

14

To-do Done

3.3

Based on **102** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Previous
Merge k sorted arrays | Set 1

Next
Find number of pairs (x, y) in an array such that x^y > y^x

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments