Radix Sort

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

Counting sort is a linear tine 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.]

17$\underline{0}$, 9$\underline{0}$, 80$\underline{2}$, $\underline{2}$, 2$\underline{4}$, 4$\underline{5}$, 7$\underline{5}$, 6$\underline{6}$

Google™ Custom Search

GeeksforGeeks
53,520 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

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.

Implementation of Radix Sort
Following is a simple C++ 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.

```
// C++ implementation of Radix Sort
#include<iostream>
using namespace std;

// A utility function to get maximum value in arr[]
```

Geometric Algorithms

FREE Infrared Automation Guidebook
The Ultimate IR Resource Guide for Automated Processes
Get the Guide
FLIR
FLIR
IR Automation Guidebook: Temperature Monitoring and Control with IR Cameras

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
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.
void countSort(int arr[], int n, int exp)
{
    int output[n]; // output array
    int i, count[10] = {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 o
    // 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
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
    // number, exp is passed. exp is 10^i where i is current digit num
    for (int exp = 1; m/exp > 0; exp *= 10)
```

```
        countSort(arr, n, exp);
}

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

// Driver program to test above functions
int main()
{
    int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};
    int n = sizeof(arr)/sizeof(arr[0]);
    radixsort(arr, n);
    print(arr, n);
    return 0;
}
```

Output:

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

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

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 4 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 7 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 32 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

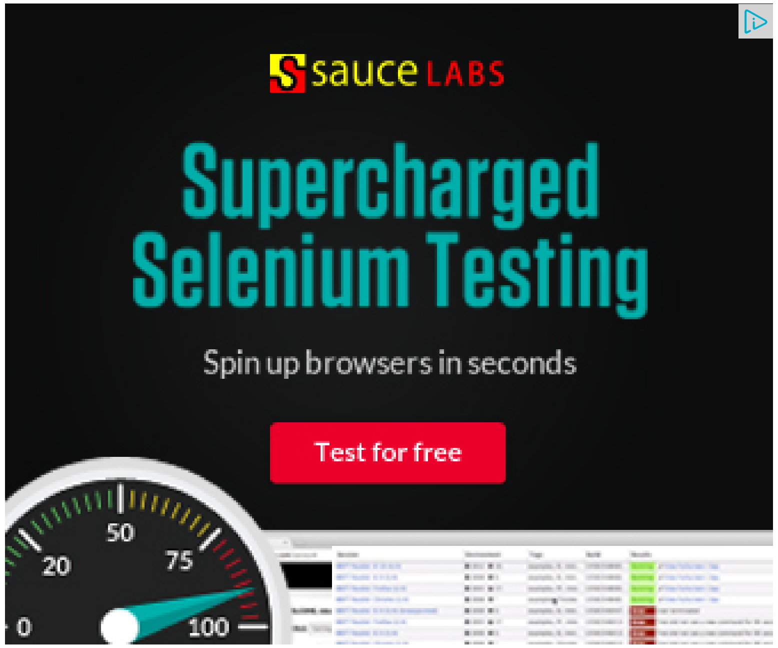
Count trailing zeroes in factorial of a number · 34 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 59 minutes ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago



Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

30 Tweet 2 1

Writing code in comment? Please use **ideone.com** and share the link here.

18 Comments GeeksforGeeks

Sort by Newest

ERROR • a month ago

```
for (i = n - 1; i >= 0; i--)
{
// YOU ARE RUNNNING OUT OF ARRAY ... YOU KNOW
output[count[ (arr[i]/exp)%10 ] - 1] = arr[i];
count[ (arr[i]/exp)%10 ]--;
}
```

YOU wrote it badli

Reply • Share ›

zzer • a month ago

here is a general version of radix sort, which you can choose any base you lik

```
#include<iostream>

#include<time.h>

using namespace std;

const int base = 16;

int get_max(int arr[],int n)

{

int max = arr[0];

for(int i = 1; i < n; i++)

if(max < arr[i])
```

see more

^ | ⌄ • Reply • Share ›

Sparsh • 3 months ago

I'm not sure if that has been said, but I think one clarification to add is that whe should be noted that the constant c in n^c should be O(1) (although I understa granted to be O(1))

A nice explanation of radix sort is given here:
http://courses.csail.mit.edu/6...

^ | ⌄ • Reply • Share ›

Prasaanth • 3 months ago

http://pastie.org/8674341#8,11

Friends , please find my recursive java implementation above and give your cc prasaanth07@gmail.com :)

^ | ⌄ • Reply • Share ›

Hitesh ➦ Prasaanth • 2 months ago

int Maxdigits=3;
It's NOT a good practice.
Your code should work for all the maximum possible input without any

^ | ⌄ • Reply • Share ›

Hitesh ➦ Prasaanth • 2 months ago

Dude! don't use built-in constructs like hashmap, etc in learning the ba

^ | ⌄ • Reply • Share ›

Guru • 5 months ago

#include<iostream>

```
#include<limits.h>
#include<vector>
#include<list>
using namespace std;
#define BASE 10

int arr[] = {23, 145, 11, 90, 115, 39, 13, 256, 47, 71};

void printArray(int arr[], int n)
{
for(int i=0; i < n; i++)
{
cout<<arr[i]<<" ";="" }="" cout<<endl;="" }="" void="" radixsort(int="" arr[],="" in
int="" digits="0;" for="" (int="" i="0;" i<n;="" i++)="" {="" if="" (="" arr[i]="">  large
largest = arr[i];
}

do {
```

see more

Reply · Share ›

Javed · 5 months ago

there is a typo....return type of countSort function should be void.

Reply · Share ›

Kartik Javed · 5 months ago

Thanks for pointing this out. We have made the return type as void.

Reply · Share ›

Allen · 6 months ago

I understand the concept behind Radix sort thanks to your post. But how do I c
made using this program?

Silent • 8 months ago

why this sorting is giving garbage value at some indices??? it's a simple coun

```
void CountSort(int arr[],int size,int exp)
{
int temp[10] = {0},b[size];
int i = 0;

while(i < size)
{
temp[(int)(arr[i] / pow(10,exp-1)) % 10]++;
i++;
}

i = 1;
while(i < 10)
{
temp[i] = temp[i-1] + 1;
i++;
}
```

see more

⌃ | ⌄ • Reply • Share ›

sumit4iit • 8 months ago

First line.
If the elements are in the range from 1 to n^2 then counting sort will take O(n^2
comparison based algorithm is going to take (n^2)lg(n^2) or O(n^2 * lg(n)) tim

You can say that it will require more space complexity than that of comparisor
has got more time complexity is incorrect.

| • Reply • Share ›

kumar ➦ sumit4iit • 6 months ago

Range != number of numbers that are to sorted.

Range means the max and min value in the array of numbers which ar

^ | ˅ • Reply • Share ›

Anonymous ➦ sumit4iit • 8 months ago

Comparison based algorithm will take O(n.log(n))

Here n is the number of elements and any comparison based sorting c

elements.

Counting sort on the other hand depends on the value of the largest ele

O(n^2)

3 ^ | ˅ • Reply • Share ›

Amandeep Sharma • 8 months ago

```
#include<iostream.h>
#include<math.h>
int main()
{
int arr[5],i,j,k=0,p=0;
int radix[5][10];
for(i=0;i<5;i++)
{
for(j=0;j<10;j++)
{
radix[i][j]=0;
}
}
cout<<"Please enter three digit no.s only :\n";
for(i=0;i<5;i++)
{
```

}

see more

• Reply • Share ›

Hiren Pandya • 8 months ago

I am sorry to bother you but I really can't understand the lines given below,

```
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 ]--;
}
```

What exactly is being done over here.. I've tried on my own to under stand by
but still no clearance of my doubt..

1 • Reply • Share ›

GeeksforGeeks Mod → Hiren Pandya • 8 months ago

The counting sort post may help you. Please see the following link.

http://www.geeksforgeeks.org/c...

• Reply • Share ›

mritunjaya gupta → GeeksforGeeks • 8 months ago

http://algogururocks.blogspot....

public static void doBalanacePartition(int[] arr) {

```
int sum = 0;

for (int i : arr) {

sum += i;

}

int num = sum / 2;

int k = num;

while (!findNumSetBySum(arr, k)) {

k++;

}

int j = num;

while (!findNumBySum(arr, j)) {

j--;

}

}
```

Reply • Share ›

Subscribe   Add Disqus to your site