The Citadel

Radix Sort

Analysis, Benchmarking, and Application

Zachary Schellinger

Data Structures and Algorithms CSCI 223

Dr. Verdicchio

Due 26 April

What is Radix Sort?

Radix sort is a non comparative sorting algorithm meaning it doesn't compare one object to another. Instead it sorts by either the most significant digit or least significant. For example you have a set of 10 digits:

13 18 48 405 39 28 74 698 22 53

It would then put the first digits in ascending or descending order depending on how you have your code set up:

{22} {13, 53} {74}, {405} {18, 48, 28, 698} {39}

Then it will start sorting by the next digit, putting it in ascending order:

{22} {13, 53} {74}, {405} {18, 48, 28, 698} {39}
{22} {13, 53} {74}, {405} {18, 48, 28, 698} {39}

{405} {13, 18,} {22, 28} {39} {48} {53} {74} {698}

Continuing this process until the sort is complete:

{405} {13, 18,} {22, 28} {39} {48} {53} {74} {698}
{405} {013,018,} {022, 028} {039} {048} {053} {074} {698}

{13, 18, 22, 28, 39, 48, 53, 74} {405} {698}

{13, 18, 22, 28, 39, 48, 53, 74, 405, 698}

And just like that the list is sorted, this can also be applied to words given each letter is assigned a numerical value. Radix sort has a $\Theta(d(n+k))$ time complexity, n being the number of keys, w being the key length and d being the number of digits in the largest element.

Where is it Used?

Radix sort can be used anywhere that data can be lexicographically sorted, i.e words and numbers. This is extremely useful when needing to sort a large number of dates, names, and age.

Types of Radix sort

LSD(Least-significant-digit)
Least significant digit radix sort, sorts from the right to left, or from the smallest place digit to the largest place digit. For example we have the set {31, 4, 74}, The sort would look at the smallest number place to begin sorting so: {31, 4, 74} → {31},{4,74} → {31},{04,74} → {04}, {31}, {74} → {4, 31, 74}.

MSD(Most-significant-digit)
Most significant digit radix sort, sorts from the left to right, or from the largest place digit to smallest place digit. For example we have the set {31, 4, 74}, the sort would look at the largest number place to begin sorting so: {31, 04, 74} → {04}, {31}, {74} → {04}, {31}, {74} → {4,31,74}.

Time complexity

Worst Case	Best Case	Average Case	Worst case space complexity
Θ(d(n+k)) / Θ(n)	Θ(d(n+k)) / Θ(n)	Θ(d(n+k)) / Θ(n)	Θ(n+k)

What are the pros and cons of Radix Sort?

Pros:

- As a non comparison sort, it has a linear running time complexity.
- Regardless of digit size, the range of sorting will never change since it can only be 0-9 or 0-25.

Cons:

- With all fast sorting algorithms, it requires more storage space for its subroutines.
- Since radix sorts by digits and letters, the application is not as flexible as other sorting algorithms.

Why is it Not Used More?

Radix sort is very quick and efficient sorting algorithm but when it comes to general purpose and usability in different applications, it loses all of its potential. Unless you need to sort big integer or char data and are willing to give up the storage space to do so, Quick Sort and Merge sort will almost always win when compared to Radix sort.

Application

System Specs:

System Model - XPS 15 7590
System Type - x64-based PC
Installed PM(RAM) - 32.0 GB
Total PM - 31.7 GB
Available PM - 19.6 GB
Total Virtual Memory - 36.5 GB
Available VM - 20.1 GB
Graphics Card - NVIDIA GeForce GTX 1650
Processor - Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 Mhz, 6 Core(s), 12 Logical Processor(s)

Benchmarking:

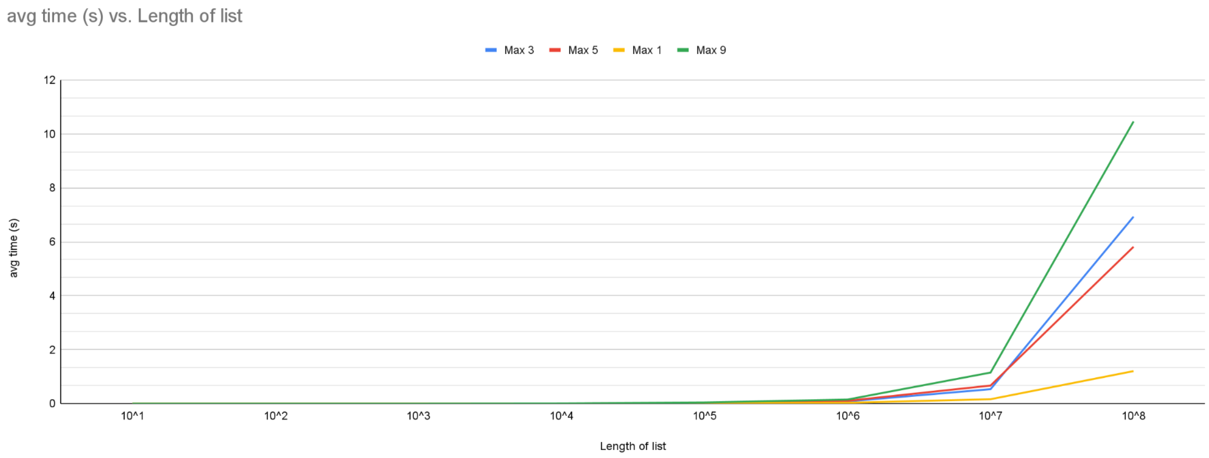
Below is a graph of 4 sets of data, benchmarking how radix sort handles a larger range of integers to run time. The control variables of this benchmark are the length of the list, the sort being used and the machine it's being run on, our independent variable being the range of length an integer can be in the list. For example it starts with the maximum range being from 1-9, giving that the name max 1, and so on and so forth all the way up to max 9 which would have numbers ranging from 1 - 999999999.

Length of list	Max world length	Time 1 (ns)	Time 2 (ns)	Time 3 (ns)	avg time (s)
10^1	1	23100	20200	27200	0.0000235
10^2	1	31400	35400	30100	0.0000323
10^3	1	360800	135700	136500	0.000211
10^4	1	2266600	1061500	2150900	0.001826333333
10^5	1	15639300	20182000	12012100	0.01594446667
10^6	1	36465000	35464600	35119800	0.03568313333
10^7	1	160768600	192753600	144602900	0.1660417
10^8	1	1256813300	1184210600	1183229300	1.2080844

Length of list	Max world length	Time 1 (ns)	Time 2 (ns)	Time 3 (ns)	avg time (s)
10^1	3	40600	27200	25200	0.0000310000
10^2	3	45200	44400	46200	0.0000452667
10^3	3	753200	1026100	684500	0.0008212667
10^4	3	4302200	4102800	3033800	0.0038129333
10^5	3	22893100	24791300	23855600	0.0238466667
10^6	3	82817200	87552700	86704300	0.0856914000
10^7	3	517150300	553899700	543459000	0.5381696667
10^8	3	5460131300	9743448700	5592449400	6.9320098000

Length of list	Max world length	Time 1 (ns)	Time 2 (ns)	Time 3 (ns)	avg time (s)
10^1	5	26900	25000	22000	0.00002463333333
10^2	5	59800	82700	61600	0.00006803333333
10^3	5	377800	395200	378000	0.0003836666667
10^4	5	3292400	3293500	4410200	0.003665366667
10^5	5	25761600	25548500	33053900	0.02812133333
10^6	5	126918400	99789900	97506400	0.1080715667
10^7	5	691924300	687362400	638224500	0.6725037333
10^8	5	6021414400	5731646600	5693495700	5.8155189

Length of list	Max world length	Time 1 (ns)	Time 2 (ns)	Time 3 (ns)	avg time (s)
10^1	9	27900	29700	29300	0.00002896666667
10^2	9	94300	95300	92300	0.00009396666667
10^3	9	716500	959600	1404900	0.001027
10^4	9	7378900	8810900	8919100	0.008369633333
10^5	9	46240300	41366500	55527600	0.04771146667
10^6	9	142785200	154000700	164747600	0.1538445
10^7	9	1214286900	1116328600	1131498200	1.1540379
10^8	9	11017936300	10223171400	10138455500	10.4598544



Analysis

Radix sort has a time complexity of Θ(d(n+k)), d being the largest number of digits in an element, n being the number of elements, and k being the largest number in the ones, tens, hundreths, etc. place. In this benchmark you can clearly see how it proves this time complexity is true, as the length and width of the list increase, the time of the sort increases. This is more drastically shown as d approaches 10^8. Overall radix sort is an extremely fast algorithm that even when computing over 100000000 integers with a varying length of over 999999999, completes it in just over 10 seconds.

Comparison

Algorithm	guarantee	random	Extra space	stable?
Mergesort	$N \lg N$	$N \lg N$	N	yes
quicksort	1.39 $N\lg N$*	1.39 $N\lg N$	$c \lg N$	no
heapsort	$2N\lg N$	$2N\lg N$	1	no
LSD sort	$2W(N+R)$	$2W(N+R)$	$2W(N+R)$	yes
MSD sort	$2W(N+R)$	$N \log_R N$	$N + DR$	

Github

https://github.com/Zmschellinger/Radix-Sort-benchmark

Sources

- https://softwareengineering.stackexchange.com/questions/77529/why-isnt-radix-sort-used-more-often
- https://www.geeksforgeeks.org/radix-sort/
- https://en.wikipedia.org/wiki/Radix_sort
- https://brilliant.org/wiki/radix-sort/
- https://www.interviewkickstart.com/learn/radix-sort-algorithm
- https://algs4.cs.princeton.edu/51radix/