Fast, stable and scalable true radix sorting

useR!, Aalborg
02 July 2015
Matt Dowle
Overview

● Released in data.table v1.9.2 (Feb 2014)
● Propose to move to base so R community can benefit with no code changes
● Find sponsor from core team with time
● Your feedback and suggestions please
2009 : Tom Short

*Tom:* I like data.table! But `setkey` is slow and I have issues with dates. Any ideas?

*Matt:* Sorry, not really. I can have a think. You're welcome to join the project?

*Tom:* Ok thanks, I will. What about using `sort.list(x, method="radix")`?

*Matt:* What's `sort.list(x, method="radix")`?


```r
> sort(c(4,2,8,7))
[1] 2 4 7 8
> order(c(4,2,8,7))
[1] 2 1 4 3
> sort.list(c(4,2,8,7))
[1] 2 1 4 3
> ?sort.list
```

"sort.list is the same [as order], using only one argument."

"x is an atomic vector"
sort.list(x) would be better named order.vector(x)

but we love it anyway ...
> x = sample(1:100000, 1e8, replace=TRUE)  #380MB
> system.time(o1 <- order(x))
  user  system elapsed
79.444   0.056  79.402
> system.time(o2 <- sort.list(x, method="radix"))
  user  system elapsed
  1.572   0.028   1.597
> identical(o1,o2)
[1] TRUE

2009-2015 : base R
Tom used `sort.list(x, method="radix")`

```
nrow(DT) = 10 million  v1.2 => v1.3
setkey(DT, a, b)  37s  5s
```

Column by column in reverse:

1. `o = 1:nrow`
2. `order o by column b`  [1\textsuperscript{st} call to `sort.list`]
3. `order o by column a`  [2\textsuperscript{nd} call to `sort.list`]

Hard to beat, even today.
R's C code for method="radix"
step 1 / 4

// find range(x) = max(x) - min(x)
for(i=0; i<n; i++) {
    if(ISNA(x[i])) continue;
    if(x[i] > xmax) xmax = x[i];
    if(x[i] < xmin) xmin = x[i];
}

range = xmax - xmin + 1;

NB: essence of code presented in these slides
R's C code for method="radix" steps 2-4 / 4

if(range > 100000) error("too large a range of values in 'x'");

long counts[ range+1 ]; // allocate
for(i=0; i<n; i++) counts[x[i] – xmin]++;
for(i=1; i<=range; i++) counts[i] += counts[i-1]; #2
for(i=n-1; i>=0; i--) ans[--counts[x[i] - xmin]] = i; #4
method = “radix”
would be better named
method = “counting”
but we love it anyway
because it is so fast
R 3.0.0

“ sort(), sort.int() and sort.list() now use radix sorting for factors of less than 100,000 levels when method is not supplied. So does order() if called with a single factor, unless na.last = NA. “

Default changed only for factors. For integers with range < 100,000 you still have to call sort.list(x, method=“radix”) manually.
[Aside] R 3.1.0

" sort.list(method = "radix") now allows negative integers (wish of PR#15644). "

PR#15644 by Matt Dowle

In step 1 (finding the range), remove one line:
if(tmp < 0) error("negative value in 'x'");
> x = sample(1:1000001, 1e8, replace=TRUE)  # 380MB
> system.time(o1 <- order(x))
  user  system elapsed
79.444  0.056  79.402
> system.time(o2 <- sort.list(x, method="radix"))
  user  system elapsed
1.572  0.028  1.597
> identical(o1,o2)
[1] TRUE
> x = sample(1:1000002, 1e8, replace=TRUE)
> system.time(o1 <- order(x))

user  system elapsed
79.416   0.044  79.361

> system.time(o2 <- sort.list(x, method="radix"))

Error in sort.list(x, method = "radix") :
  too large a range of values in 'x'
data.table:::forderv(x)

> x = sample(1:1000002, 1e8, replace=TRUE)
> system.time(o1 <- order(x))
  user  system elapsed
  79.416  0.044  79.361
> system.time(o2 <- data.table:::forderv(x))
  user  system elapsed
  1.664  0.060  1.722
> identical(o1,o2)
[1] TRUE
Scaling up now possible

```r
> x = sample(1:1e6, 1e9, replace=TRUE)
> system.time(o2 <- data.table:::forderv(x))
  user  system elapsed
18.716   0.288  18.982
> system.time(o1 <- order(v))
Over 20 mins then I stopped it
```
true radix sorting

To illustrate, consider these 2 numbers as 4 *columns* of bytes, each with range 256:

```
705788748
```

```
  25
00101010 00010001 01111011 01001100
00000000 00000000 00000000 00010111
```

```
  42
  0
```

```
  17
  0
```

```
  123
  0
```

```
  76
  25
```
Proceed just like Tom did on columns in reverse order, but on bytes within the integer
numeric

Terdiman, 2000

http://codercorner.com/RadixSortRevisited.htm

Herf, 2001

http://stereopsis.com/radix.html

Arun Srinivasan implemented $\texttt{forder()}$ in data.table entirely in C for integer, character and double

Matt Dowle changed from LSD (backwards) to MSD (forwards) for cache efficiency and to benefit from (partially) sorted data inputs
> x = runif(500e6)  # unique this time
> system.time(data.table:::forderv(x))

MacBook Pro 2.8GHz Intel Core i7 16GB
R 3.1.3   data.table 1.9.4
Miscellaneous

- `setNumericRounding(2|1|0)`
- CHARXP are sorted by pointer value to get uniques, then uniques are sorted by forwards radix on the character string
- Endian dependent hence QEMU emulation of PowerPC before release to CRAN
- We appreciate CRAN's Solaris Sparc – it's proxy for other big endian machines
- Partial sorting, median, quantiles
- MSD radix sort is parallelizable
Thank you

Please try out data.table:::forderv()

Questions / suggestions?