Input Structure

date.in
[number of tests] [n_1] [$nmax_1$] [n_2] [$nmax_2$] ..... [$n_{[number\ of\ tests]}$] [$nmax_{[number\ of\ tests]}$]

n_i = number of numbers that are randomly generated for the i-th test

$nmax_i$ = the maximum value of the randomly generated numbers for the i-th test; the numbers generated are in range $[0, nmax_i]$

Output Structure

date.out
Test [i] [sorting_algorithm$_1$] [time_in_seconds$_1$]s [was_sorted$_1$] [sorting_algorithm$_2$] [time_in_seconds$_2$]s [was_sorted$_2$] ..... [sorting_algorithm$_n$] [time_in_seconds$_n$]s [was_sorted$_n$] ----------

n = number of tests

i = current test; i is a number in range $[0, n]$

$[sorting\_algorithm_i]$ = one of {TimSort, MergeSort, RadixSort, ShellSort, CountingSort}

$[was\_sorted_i]$ = 1 if the numbers are sorted after the i-th sorting algorithm, else 0

Algorithms Time Complexity

Tim Sort – O(n log n)

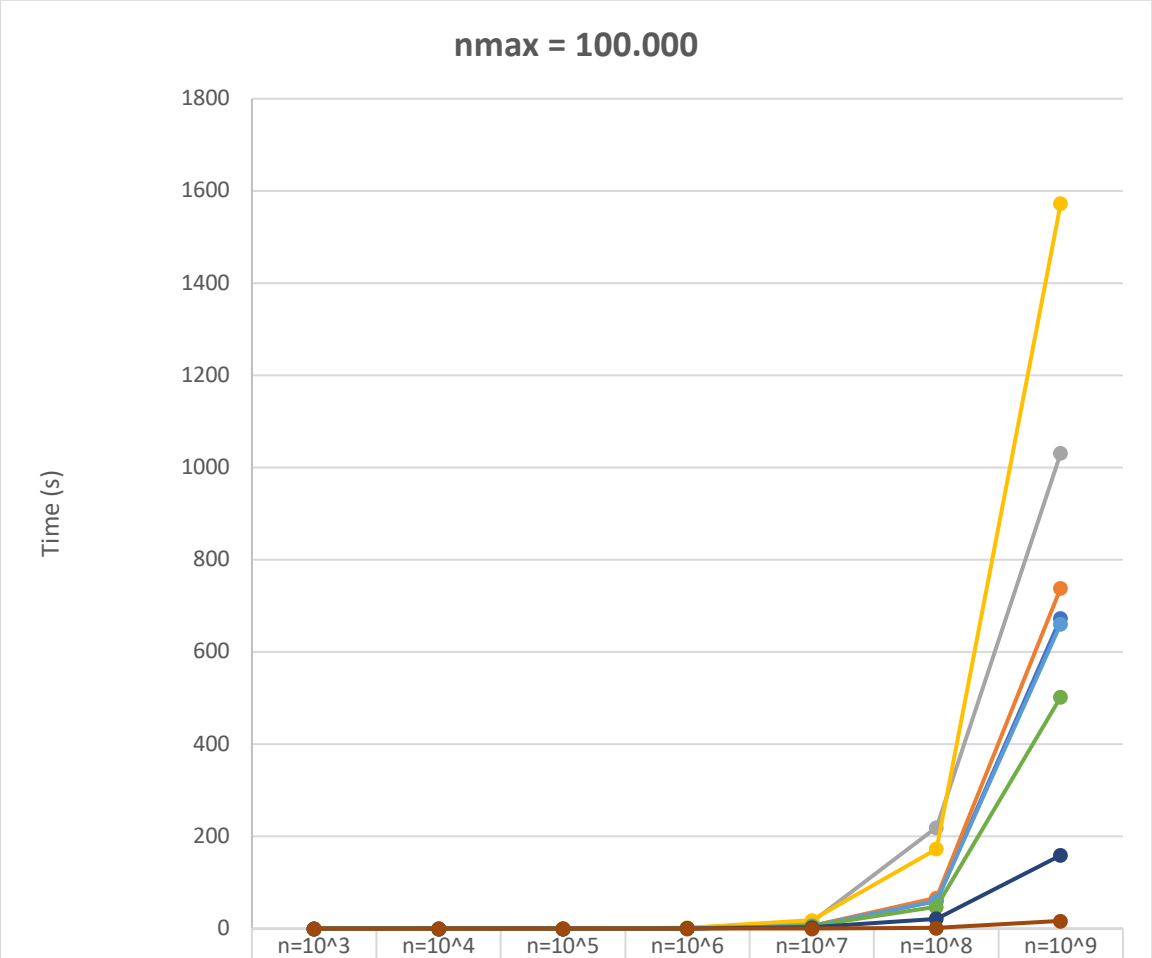
Merge Sort – O(n log n)

Shell Sort – O(n log n)

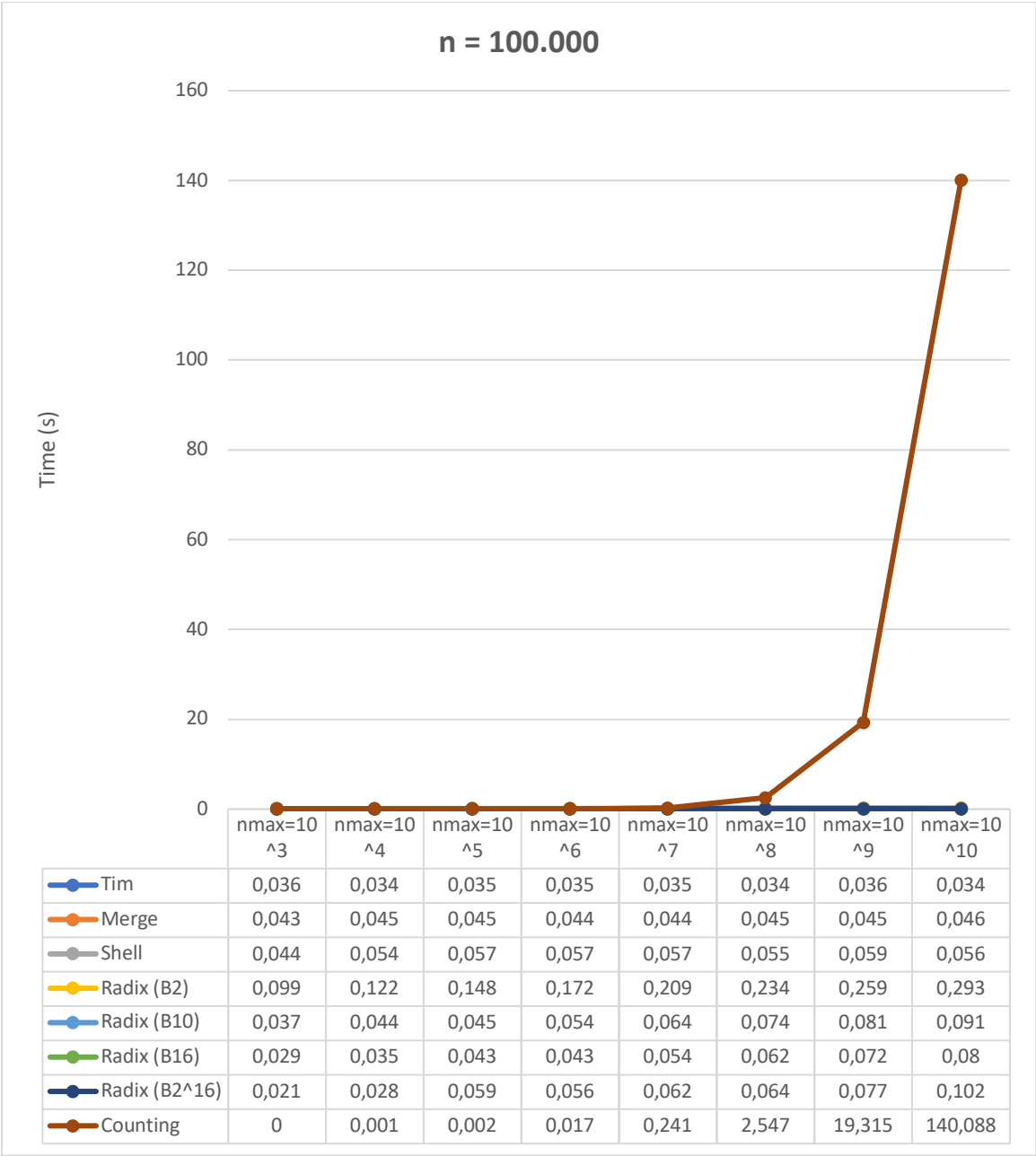
Radix Sort – O(n log nmax)

Counting Sort – O(n + nmax)

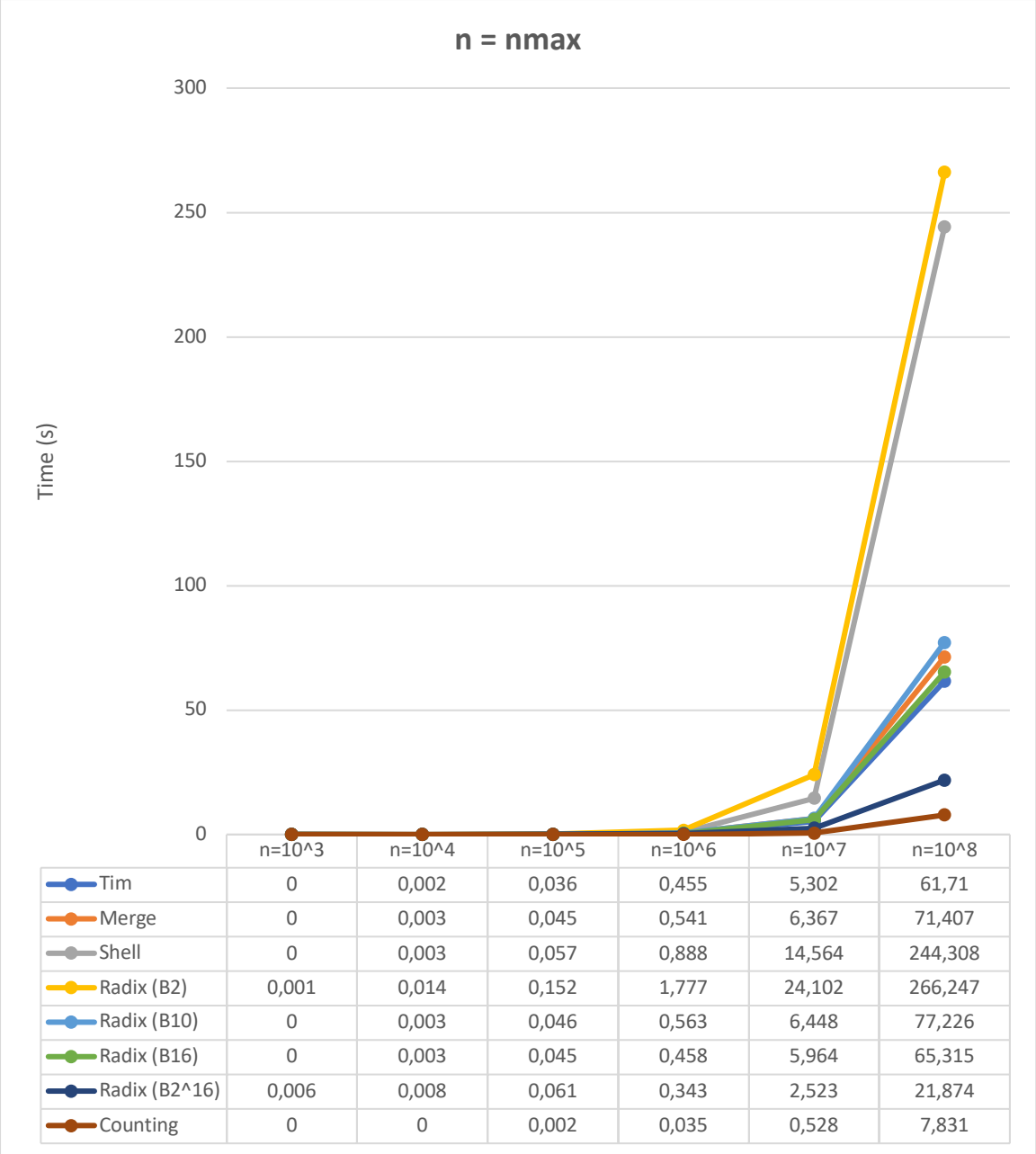
Execution Time Comparison



	n=10^3	n=10^4	n=10^5	n=10^6	n=10^7	n=10^8	n=10^9
Tim	0	0,002	0,037	0,455	5	59,96	673,521
Merge	0	0,003	0,045	0,59	6,348	66,517	738,878
Shell	0	0,003	0,06	0,931	14,218	219,143	1031,1
Radix (B2)	0,001	0,015	0,149	1,542	17,942	172,539	1573,05
Radix (B10)	0	0,004	0,054	0,61	6,012	59,344	660,812
Radix (B16)	0	0,004	0,045	0,454	5,017	47,258	502,248
Radix (B2^16)	0,012	0,015	0,058	0,345	2,573	21,259	158,892
Counting	0,001	0,001	0,002	0,014	0,109	0,952	16,653



	nmax=10^3	nmax=10^4	nmax=10^5	nmax=10^6	nmax=10^7	nmax=10^8	nmax=10^9	nmax=10^10
Tim	0,036	0,034	0,035	0,035	0,035	0,034	0,036	0,034
Merge	0,043	0,045	0,045	0,044	0,044	0,045	0,045	0,046
Shell	0,044	0,054	0,057	0,057	0,057	0,055	0,059	0,056
Radix (B2)	0,099	0,122	0,148	0,172	0,209	0,234	0,259	0,293
Radix (B10)	0,037	0,044	0,045	0,054	0,064	0,074	0,081	0,091
Radix (B16)	0,029	0,035	0,043	0,043	0,054	0,062	0,072	0,08
Radix (B2^16)	0,021	0,028	0,059	0,056	0,062	0,064	0,077	0,102
Counting	0	0,001	0,002	0,017	0,241	2,547	19,315	140,088



	n=10^3	n=10^4	n=10^5	n=10^6	n=10^7	n=10^8
Tim	0	0,002	0,036	0,455	5,302	61,71
Merge	0	0,003	0,045	0,541	6,367	71,407
Shell	0	0,003	0,057	0,888	14,564	244,308
Radix (B2)	0,001	0,014	0,152	1,777	24,102	266,247
Radix (B10)	0	0,003	0,046	0,563	6,448	77,226
Radix (B16)	0	0,003	0,045	0,458	5,964	65,315
Radix (B2^16)	0,006	0,008	0,061	0,343	2,523	21,874
Counting	0	0	0,002	0,035	0,528	7,831

Conclusions

1. Counting Sort is the most time efficient sorting algorithm, if `n = nmax`.
2. The base of Radix Sort needs to be higher for bigger numbers for the algorithm to be efficient.
3. Radix Sort is the most efficient comparison based sorting algorithm for integers.
4. Tim Sort seems to be the most efficient comparison based algorithm for any type of numbers. (not tested for rational numbers)