My answers are based on the below data AND some code analysis.

---------------------------------------------------------------

Sorting Algorithm: insertion\_sort on 1000 items.

---------------------------------------------------------------

1 function calls in 0.022 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.022 0.022 0.022 0.022 profile:0(insertion\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: insertion\_sort on 2000 items.

---------------------------------------------------------------

1 function calls in 0.044 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.044 0.044 0.044 0.044 profile:0(insertion\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: insertion\_sort on 4000 items.

---------------------------------------------------------------

1 function calls in 0.121 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.121 0.121 0.121 0.121 profile:0(insertion\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: insertion\_sort on 32000 items.

---------------------------------------------------------------

1 function calls in 2.834 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 2.834 2.834 2.834 2.834 profile:0(insertion\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: insertion\_sort on 64000 items.

---------------------------------------------------------------

1 function calls in 4.979 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 4.979 4.979 4.979 4.979 profile:0(insertion\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: insertion\_sort on 128000 items.

---------------------------------------------------------------

1 function calls in 20.598 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 20.598 20.598 20.598 20.598 profile:0(insertion\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: quicksort on 1000 items.

---------------------------------------------------------------

1 function calls in 0.004 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.004 0.004 0.004 0.004 profile:0(quicksort(L))

---------------------------------------------------------------

Sorting Algorithm: quicksort on 2000 items.

---------------------------------------------------------------

1 function calls in 0.009 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.009 0.009 0.009 0.009 profile:0(quicksort(L))

---------------------------------------------------------------

Sorting Algorithm: quicksort on 4000 items.

---------------------------------------------------------------

1 function calls in 0.019 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.019 0.019 0.019 0.019 profile:0(quicksort(L))

---------------------------------------------------------------

Sorting Algorithm: quicksort on 32000 items.

---------------------------------------------------------------

1 function calls in 0.165 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.165 0.165 0.165 0.165 profile:0(quicksort(L))

---------------------------------------------------------------

Sorting Algorithm: quicksort on 64000 items.

---------------------------------------------------------------

1 function calls in 0.351 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.351 0.351 0.351 0.351 profile:0(quicksort(L))

---------------------------------------------------------------

Sorting Algorithm: quicksort on 128000 items.

---------------------------------------------------------------

1 function calls in 0.745 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.745 0.745 0.745 0.745 profile:0(quicksort(L))

---------------------------------------------------------------

Sorting Algorithm: builtin\_sort on 1000 items.

---------------------------------------------------------------

1 function calls in 0.001 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.001 0.001 0.001 0.001 profile:0(builtin\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: builtin\_sort on 2000 items.

---------------------------------------------------------------

1 function calls in 0.001 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.001 0.001 0.001 0.001 profile:0(builtin\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: builtin\_sort on 4000 items.

---------------------------------------------------------------

1 function calls in 0.002 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.002 0.002 0.002 0.002 profile:0(builtin\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: builtin\_sort on 32000 items.

---------------------------------------------------------------

1 function calls in 0.020 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.020 0.020 0.020 0.020 profile:0(builtin\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: builtin\_sort on 64000 items.

---------------------------------------------------------------

1 function calls in 0.043 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.043 0.043 0.043 0.043 profile:0(builtin\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: builtin\_sort on 128000 items.

---------------------------------------------------------------

1 function calls in 0.109 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

1 0.109 0.109 0.109 0.109 profile:0(builtin\_sort(L))

0 0.000 0.000 profile:0(profiler)

---------------------------------------------------------------

Sorting Algorithm: radix\_sort on 1000 items.

---------------------------------------------------------------

1 function calls in 0.009 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.009 0.009 0.009 0.009 profile:0(radix\_sort(L))

---------------------------------------------------------------

Sorting Algorithm: radix\_sort on 2000 items.

---------------------------------------------------------------

1 function calls in 0.024 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.024 0.024 0.024 0.024 profile:0(radix\_sort(L))

---------------------------------------------------------------

Sorting Algorithm: radix\_sort on 4000 items.

---------------------------------------------------------------

1 function calls in 0.047 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.047 0.047 0.047 0.047 profile:0(radix\_sort(L))

---------------------------------------------------------------

Sorting Algorithm: radix\_sort on 32000 items.

---------------------------------------------------------------

1 function calls in 0.662 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.662 0.662 0.662 0.662 profile:0(radix\_sort(L))

---------------------------------------------------------------

Sorting Algorithm: radix\_sort on 64000 items.

---------------------------------------------------------------

1 function calls in 1.492 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 1.492 1.492 1.492 1.492 profile:0(radix\_sort(L))

---------------------------------------------------------------

Sorting Algorithm: radix\_sort on 128000 items.

---------------------------------------------------------------

1 function calls in 7.779 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 7.779 7.779 7.779 7.779 profile:0(radix\_sort(L))

`~~~~~~~~~~~Answers start here~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Insertion sort | Quicksort | Built in sort | Radix sort |
| 1000 | 0.022 | 0.004 | 0.001 | 0.009 |
| 2000 | 0.044 | 0.009 | 0.001 | 0.024 |
| 4000 | 0.121 | 0.019 | 0.002 | 0.047 |
| 32000 | 2.834 | 0.165 | 0.020 | 0.662 |
| 64000 | 4.979 | 0.351 | 0.043 | 1.492 |
| 128000 | 20.598 | 0.745 | 0.109 | 7.779 |

1) The insertion sort is not O (n2) from the data; the searching algorithm for the swap is not linear (scanning one by one through the whole sorted list) but rather logarithmic (using binary search which is log2 n to find the right place in the whole sorted list) so it is O(n log n) instead. At 3200, 64000, 128000, there is an irregular growth of the sorting time. They seem to be a linear growth, with an multiplication factor, this factor is log n, which is logarithmic and depends on the size of the list.

2) Radix sort is sometimes faster than insertion sort but never faster than quicksort or the built-in sort. I have tried changing the bases of radix sort to 100, 10000, 1000000 but they have not managed to sort faster than quicksort or built-in sort. Radix sort is slower than insertion sort when the bases are relatively small.

Radix sort is equivalent to comparison-based sorts in the sense that both type of sorts depends on the properties of a given array, and hence neither not necessity always better the other in every situation. This is because the efficiency of radix sorts is based on the length of the longest value, and the also length of other data values. Comparison based instead is based on the position of the value in the list relative to another value, and the overall shape (reverse, nearly-sorted, random) of the list.

4) I needed additional queues, because the current bins required can only work for non-negative integers. As when enqueuing the negative values directly to the current bins, the order they are placed in is not differentiated (and hard to make the bin tell the difference without an inefficient algorithm), when it should be negative values all at the front and positive values at back.

6)

---------------------------------------------------------------

Sorting Algorithm: sortbase2 on 1000 items.

---------------------------------------------------------------

1 function calls in 0.070 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.070 0.070 0.070 0.070 profile:0(sortbase2(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase2 on 2000 items.

---------------------------------------------------------------

1 function calls in 0.140 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.140 0.140 0.140 0.140 profile:0(sortbase2(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase2 on 4000 items.

---------------------------------------------------------------

1 function calls in 0.337 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.337 0.337 0.337 0.337 profile:0(sortbase2(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase2 on 32000 items.

---------------------------------------------------------------

1 function calls in 4.442 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 4.442 4.442 4.442 4.442 profile:0(sortbase2(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase2 on 64000 items.

---------------------------------------------------------------

1 function calls in 13.310 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 13.310 13.310 13.310 13.310 profile:0(sortbase2(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase2 on 128000 items.

---------------------------------------------------------------

1 function calls in 49.056 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 49.056 49.056 49.056 49.056 profile:0(sortbase2(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase10 on 1000 items.

---------------------------------------------------------------

1 function calls in 0.015 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.015 0.015 0.015 0.015 profile:0(sortbase10(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase10 on 2000 items.

---------------------------------------------------------------

1 function calls in 0.039 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.039 0.039 0.039 0.039 profile:0(sortbase10(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase10 on 4000 items.

---------------------------------------------------------------

1 function calls in 0.048 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.048 0.048 0.048 0.048 profile:0(sortbase10(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase10 on 32000 items.

---------------------------------------------------------------

1 function calls in 0.869 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.869 0.869 0.869 0.869 profile:0(sortbase10(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase10 on 64000 items.

---------------------------------------------------------------

1 function calls in 1.702 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 1.702 1.702 1.702 1.702 profile:0(sortbase10(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase10 on 128000 items.

---------------------------------------------------------------

1 function calls in 8.340 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 8.340 8.340 8.340 8.340 profile:0(sortbase10(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase20 on 1000 items.

---------------------------------------------------------------

1 function calls in 0.009 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.009 0.009 0.009 0.009 profile:0(sortbase20(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase20 on 2000 items.

---------------------------------------------------------------

1 function calls in 0.017 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.017 0.017 0.017 0.017 profile:0(sortbase20(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase20 on 4000 items.

---------------------------------------------------------------

1 function calls in 0.036 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.036 0.036 0.036 0.036 profile:0(sortbase20(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase20 on 32000 items.

---------------------------------------------------------------

1 function calls in 0.467 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 0.467 0.467 0.467 0.467 profile:0(sortbase20(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase20 on 64000 items.

---------------------------------------------------------------

1 function calls in 1.084 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 1.084 1.084 1.084 1.084 profile:0(sortbase20(L))

---------------------------------------------------------------

Sorting Algorithm: sortbase20 on 128000 items.

---------------------------------------------------------------

1 function calls in 2.358 seconds

Ordered by: standard name

ncalls tottime percall cumtime percall filename:lineno(function)

0 0.000 0.000 profile:0(profiler)

1 2.358 2.358 2.358 2.358 profile:0(sortbase20(L))

|  |  |  |  |
| --- | --- | --- | --- |
|  | Base2 | Base 10 | Base 20 |
| 1000 | 0.070 | 0.015 | 0.009 |
| 2000 | 0.140 | 0.039 | 0.017 |
| 4000 | 0.337 | 0.048 | 0.036 |
| 32000 | 4.442 | 0.869 | 0.467 |
| 64000 | 13.310 | 1.702 | 1.084 |
| 128000 | 49.056 | 8.340 | 2.358 |

From the data, as base (the radix), the run time significantly decreases. This suggests that the most time-consuming process is removing items from the queues. The most likely reason that most people don’t use values greater than base 10 is that it is hard to initially code a radix\_sort of a radix sort other than base 10.

\*I have tested the code by using up to a base of 1,000,000(not shown here) and the run time actually INCREASES after a certain point. It seems there is a “threshold” of bases where the run time won’t get any faster and instead get slower. So on a x vs y graph where x is base #, and y is run time, a parabolic curve pointing up can be expected when dealing with massive numbers of x. By extrapolating from this fact, a reason why base 10 can be used is that it can be one of these “threshold” values for certain type of radix sort algorithms. (My personal algorithm may be inefficient)