FELIPE FERNANDEZ CRONENBOLD

For the decision of choosing M, I used the powers of 2 because an exponential function is a good way to find its median value.

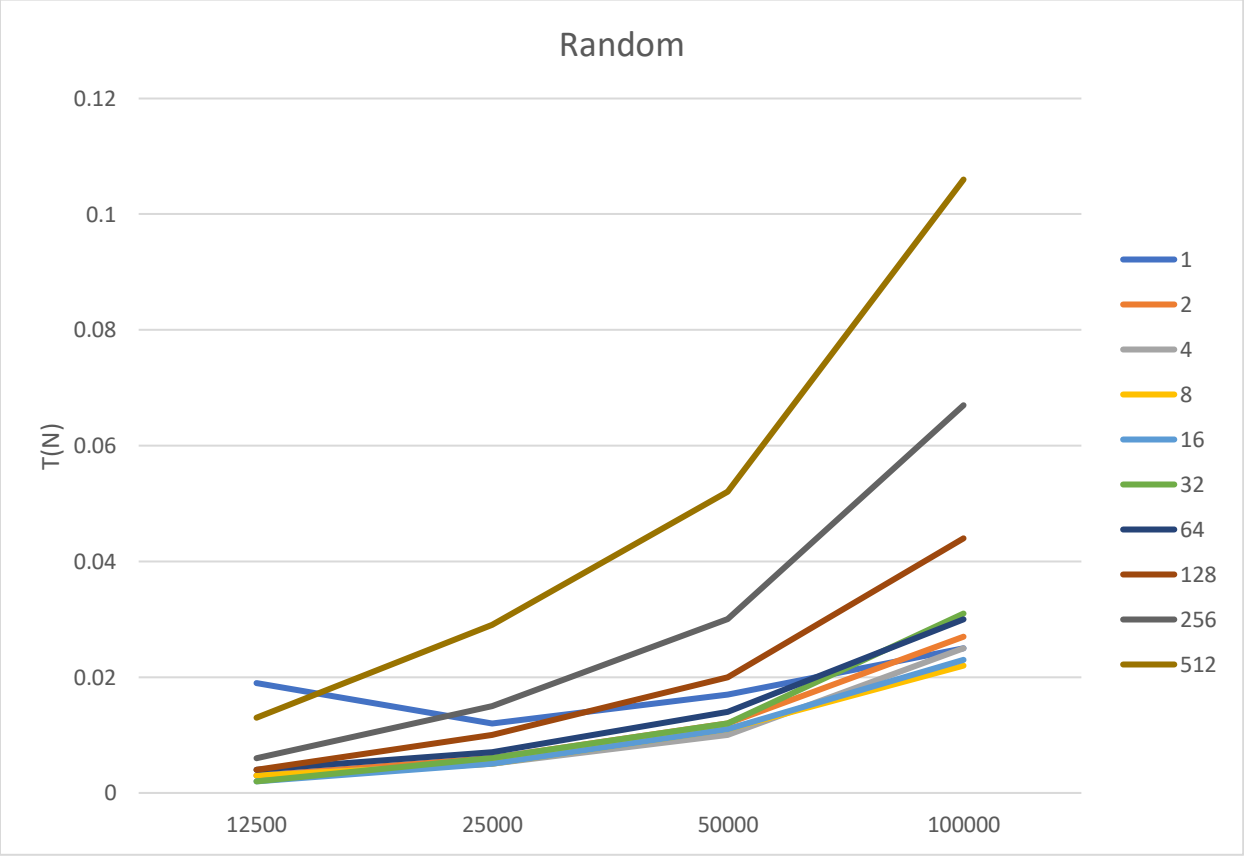


Chart Time(N, M)

	M	1	2	4	8	16	32	64	128	256	512
N											
12500		0.019	0.003	0.003	0.003	0.002	0.002	0.004	0.004	0.006	0.013
25000		0.012	0.006	0.005	0.005	0.005	0.006	0.007	0.01	0.015	0.029
50000		0.017	0.012	0.01	0.011	0.011	0.012	0.014	0.02	0.03	0.052
100000		0.025	0.027	0.025	0.022	0.023	0.031	0.03	0.044	0.067	0.106

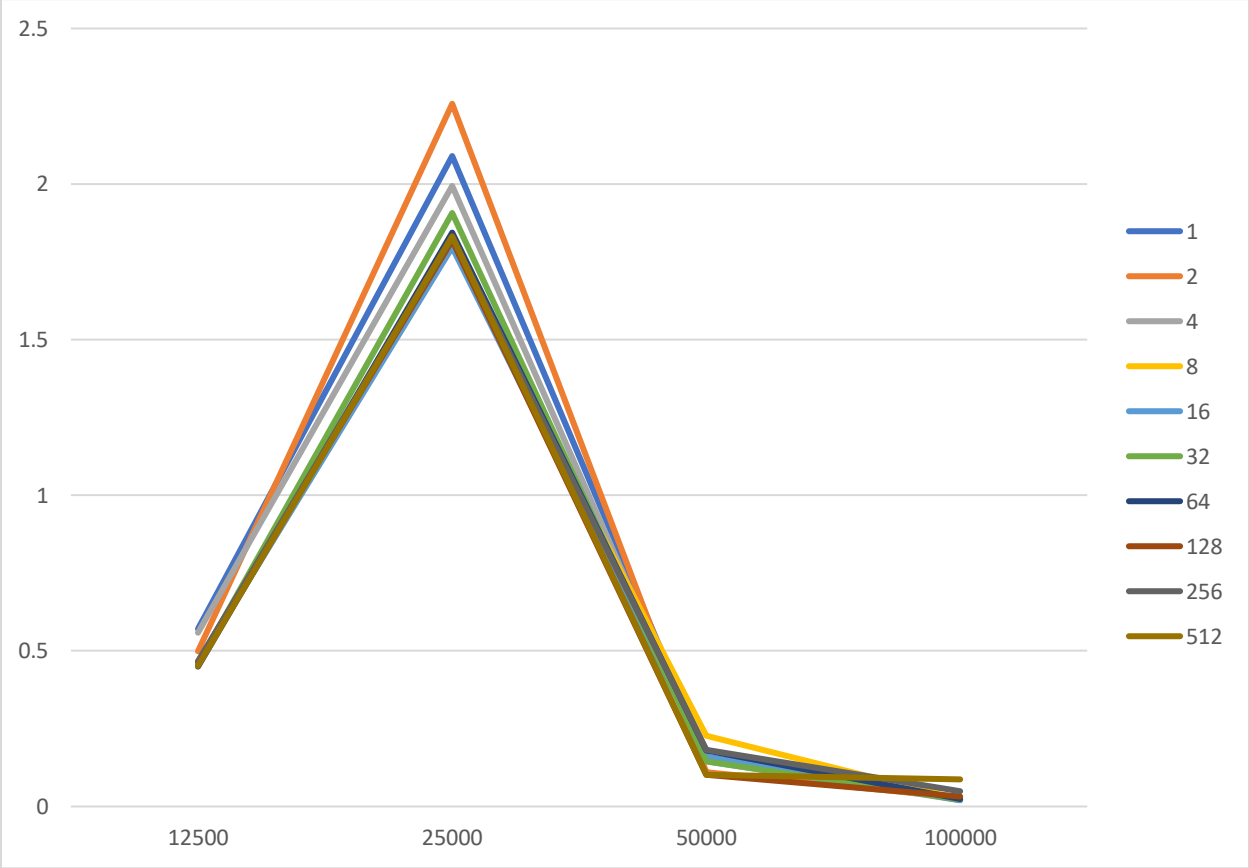
The best T(N) here is registered between series:

a- 16
b- 8
c- 4

The serie “yellow” it’s the one that gives the best time at higher Ns but it has a worst performance in lows n’s so for a random probe I would say that its between 4 and 16 (grey or light blue) in which case the graph clearly shows that at higher Ns blue would work better.

Using values greater than 128 would be controversial because sorting 128 items on the worst case is at least n^2 so T(N) order is at its worst-case c*16384 which is a risk not worth it for the benefits, so we are going to limit our analysis only to range [1:64] without analyzing other than T(n). swaps and comparisons are not taken into consideration because a hybrid quicksort has always more comparisons than a quicksort so the higher the M the more comparisons and swaps made but this doesn't mean a performance decrease that's why times are analyzed.

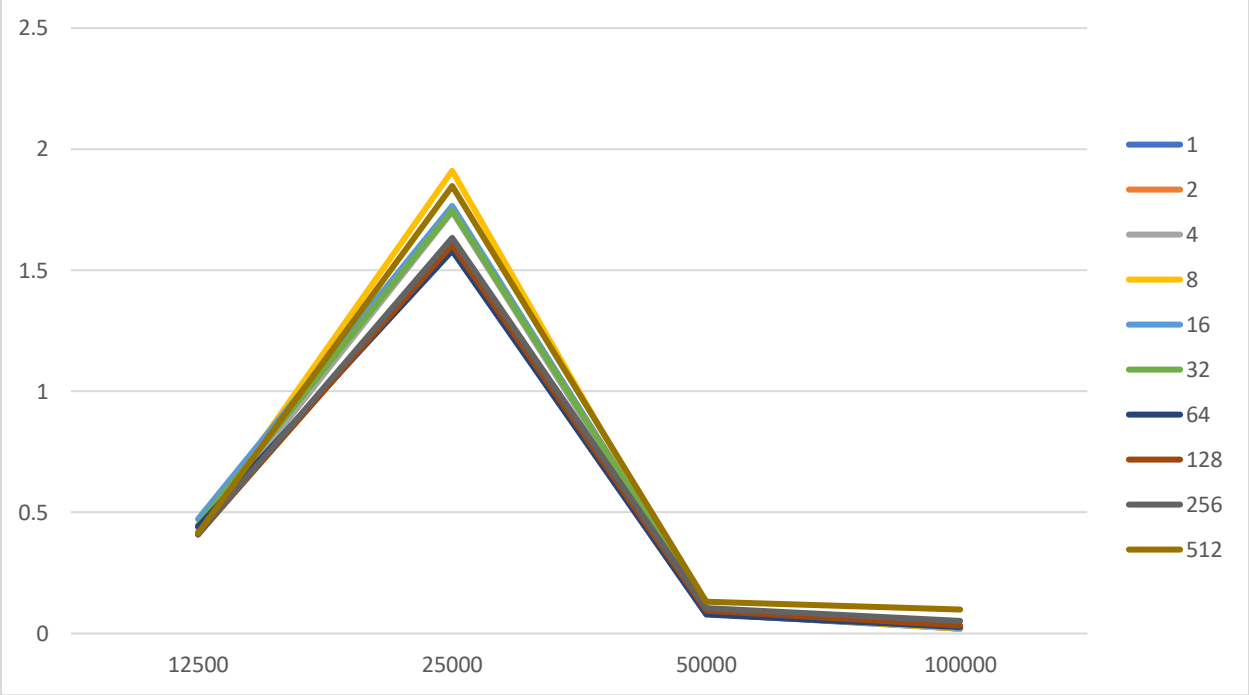
So looking into ascending best-case for insertion sort:



Sum of time	Column Labels									
Row Labels	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
12500	0.571	0.499	0.558	0.462	0.46	0.457	0.449	0.466	0.465	0.451
25000	2.09	2.258	1.994	1.799	1.797	1.907	1.844	1.819	1.828	1.832
50000	0.16	0.109	0.144	0.227	0.159	0.145	0.18	0.102	0.183	0.102
100000	0.021	0.028	0.02	0.023	0.02	0.022	0.024	0.032	0.049	0.087
Grand Total	**2.842**	**2.894**	**2.716**	**2.511**	**2.436**	**2.531**	**2.497**	**2.419**	**2.525**	**2.472**

Looking to this chart we are now seeing some great performance at some high M values, but we are still seeing that M = 16 it is finding its way between these values being second with a difference of 0.017ms which is worth considering global functioning aspects like random for instance. So, looking at this graph and using the previous information M=16 is still a viable option.

Finally evaluating the descending chart:



Row Labels	1	2	4	8	16	32	64	128	256	512
12500	0.41	0.409	0.412	0.422	0.472	0.447	0.441	0.412	0.41	0.413
25000	1.756	1.608	1.742	1.911	1.766	1.747	1.581	1.605	1.634	1.848
50000	0.08	0.091	0.086	0.086	0.097	0.101	0.078	0.093	0.104	0.132
100000	0.02	0.02	0.02	0.02	0.022	0.027	0.027	0.033	0.052	0.099
Grand Total	**2.266**	**2.128**	**2.26**	**2.439**	**2.357**	**2.322**	**2.127**	**2.143**	**2.2**	**2.492**

Looking at this chart we can still see M=16 with a great T(N) even when all function deviations are maximum in N= 25 000 when the difference between M = 16 and lowest time in N =25 000 M = 64 is:

= |1.581-1.747|=0.166 ms which is a nice difference for a best global performance.
this same M will be used in the QuickSorterMedianThree Algorithm.