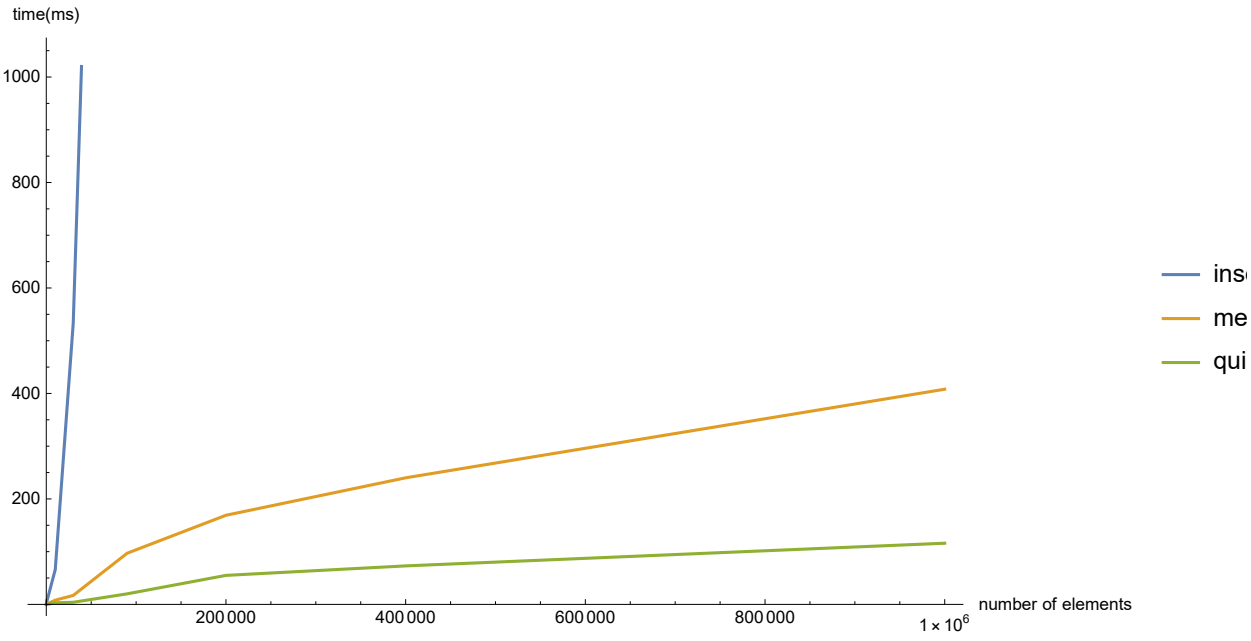
Uppgift 4

In[ ]:=
```
insertSortTimes =
  {{100, 0}, {1000, 8}, {10 000, 66}, {30 000, 533}, {90 000, 3705}, {200 000, 18 079}};
mergeSortTimes = {{100, 0}, {1000, 1}, {10 000, 8}, {30 000, 17},
   {90 000, 97}, {200 000, 169}, {400 000, 240}, {1 000 000, 408}};
quickSortTimes = {{100, 0}, {1000, 0}, {10 000, 3}, {30 000, 4},
   {90 000, 20}, {200 000, 55}, {400 000, 73}, {1 000 000, 116}};
ListLinePlot[{insertSortTimes, mergeSortTimes, quickSortTimes},
 AxesLabel → {"number of elements", "time(ms)"},
 PlotLegends → {"insertion sort", "merge sort", "quick sort"}]
```

Out[ ]=



DISCUSSION:

We see clearly that **insertion sort** take exponentially more time when increasing the amount of elements to be sorted. We see that both **merge sort** and **quick sort** grows in a logarithmic pattern, with quick sort growing remarkably slower.

*Insertion sort is obviously the slowest sorting algorithm of the three tested, but the algorithm is also simpler to understand/implement making it a solid choice of sorting algorithm if handling small amounts of data.

*Merge sort definitely has its advantage over insertion sort when comparing speed. However, if falls short when compared to quick sort, being faster in all scenarios of this lab.

*Quick sort is the clear winner in this lab, it is a really fast sorting algorithm that can sort a large amount of data within reasonable time. The only possible con is that it is a bit more complex to understand and more difficult to implement, which might make it less neccesary if working with small amounts of data.