* selection sort:
repeatedly find the next smallest element and put it in its place

* insertion sort:
repeatedly place the next element in position among the elements already sorted

* heap sort:
This can be done by making a copy of the list into a separate heap. However, it's also possible to do it without having to make a copy...
Idea: use a max-heap.
* merge sort:

  given list \( L \):
  sort the first half of \( L \), recursively
  sort the second half of \( L \), recursively

  \[
  L : \quad \begin{array}{c}
  \quad \text{sort} \\
  1, 7, 13, 24, 40 \\
  \end{array}
  \begin{array}{c}
  \quad \text{sort} \\
  7, 8, 20, 21, 30 \\
  \end{array}
  \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad 1 \ 2 \ 5 \ 7 \ 13 \ \ldots
  \\
  \end{array}
  \end{array}
  \end{array}
  \]

  quicksort:

  \[
  L : \quad \begin{array}{c}
  \quad \text{partition} \\
  \quad \begin{array}{c}
  \quad \text{sort} \\
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad 1 \ p \ \ldots
  \\
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \begin{array}{c}
  \quad \begin{array}{c}
  \quad \text{sort} \\
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad 1 \ p \ \ldots
  \\
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad \begin{array}{c}
  \quad 1 \ p \ \ldots
  \\
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \end{array}
  \]