```
Timing summary:
   build initial array & SelectionSort: 0 second(s)
   refill array & MergeSort, 200 times: 1 second(s)

➜ gprof ./a.out | more
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 54.64      0.53     0.53  4999800     0.00     0.00  Merge
 40.21      0.92     0.39        1   390.00   390.00  SelectionSort
  5.15      0.97     0.05      200     0.25     2.90  MergeSort

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
--More--
```

```
                     Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) for 1.03% of 0.97 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    0.97                 main [1]
                0.05    0.53     200/200         MergeSort [2]
                0.39    0.00       1/1           SelectionSort [4]
-----------------------------------------------
                             9999600             MergeSort [2]
                0.05    0.53     200/200         main [1]
[2]     59.8    0.05    0.53     200+9999600 MergeSort [2]
                0.53    0.00 4999800/4999800     Merge [3]
                             9999600             MergeSort [2]
-----------------------------------------------
                0.53    0.00 4999800/4999800     MergeSort [2]
[3]     54.6    0.53    0.00 4999800         Merge [3]
-----------------------------------------------
                0.39    0.00       1/1           main [1]
[4]     40.2    0.39    0.00       1         SelectionSort [4]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
 index number at the left hand margin lists the current function.
 The lines above it list the functions that called this function,
```

25000

```
Timing summary:
   build initial array & SelectionSort: 3 second(s)
   refill array & MergeSort, 200 times: 1 second(s)

➜ gprof ./a.out | more
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 58.01      2.21     2.21        1     2.21     2.21  SelectionSort
 40.68      3.76     1.55  9999800     0.00     0.00  Merge
  1.31      3.81     0.05      200     0.00     0.01  MergeSort

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
--More--
```

```
^L
                     Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) for 0.26% of 3.81 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    3.81                 main [1]
                2.21    0.00       1/1           SelectionSort [2]
                0.05    1.55     200/200         MergeSort [3]
-----------------------------------------------
                2.21    0.00       1/1           main [1]
[2]     58.0    2.21    0.00       1         SelectionSort [2]
-----------------------------------------------
                           19999600             MergeSort [3]
                0.05    1.55     200/200         main [1]
[3]     42.0    0.05    1.55     200+19999600 MergeSort [3]
                1.55    0.00 9999800/9999800     Merge [4]
                           19999600             MergeSort [3]
-----------------------------------------------
                1.55    0.00 9999800/9999800     MergeSort [3]
[4]     40.7    1.55    0.00 9999800         Merge [4]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
 index number at the left hand margin lists the current function.
```

50000

```
➜ gprof ./a.out | more
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 68.59      6.31     6.31        1     6.31     6.31  SelectionSort
 30.87      9.15     2.84 19999800     0.00     0.00  Merge
  0.33      9.18     0.03                             main
  0.22      9.20     0.02      200     0.00     0.01  MergeSort

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

--More--
```


```
➜ gprof ./a.out | more
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 68.59      6.31     6.31        1     6.31     6.31  SelectionSort
 30.87      9.15     2.84 19999800     0.00     0.00  Merge
  0.33      9.18     0.03                             main
  0.22      9.20     0.02      200     0.00     0.01  MergeSort

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

--More--
```


10000

Time complexity differences

- For the list of 100000 elements, selection sort was called only once, taking up 68.59% of program runtime, while mergesort was called 200 times in less time.
- The biggest reason for this time difference is how other algorithms are implemented. Selection sort uses nested for loops, so the time complexity is always O(n^2), while mergesort recursively sorts the left and right sides of the list, so the time complexity is O(n log n)

Pros and cons

- Mergesort will always be faster because it is implemented recursively instead of with nested loops.
- The main pro of selection sort is how easily it can be implemented.
- Mergesort is better for longer lists, but it is harder to implement

Merge was called 19999800 times for a list with 100000 elements, and merge was called 200 times.
This matches the expectation since merge is called when a sorted array needs to be merged back into a larger sorted array.
Gprof reports the number of times a function is entered, which doesn't work for recursive functions since they call themselves within itself.