**Part 1 – Performance of Searching Routines**

The following questions are taken from *A laboratory Course in C++ Data Structures*, by Roberge, Brandle and Whittington:

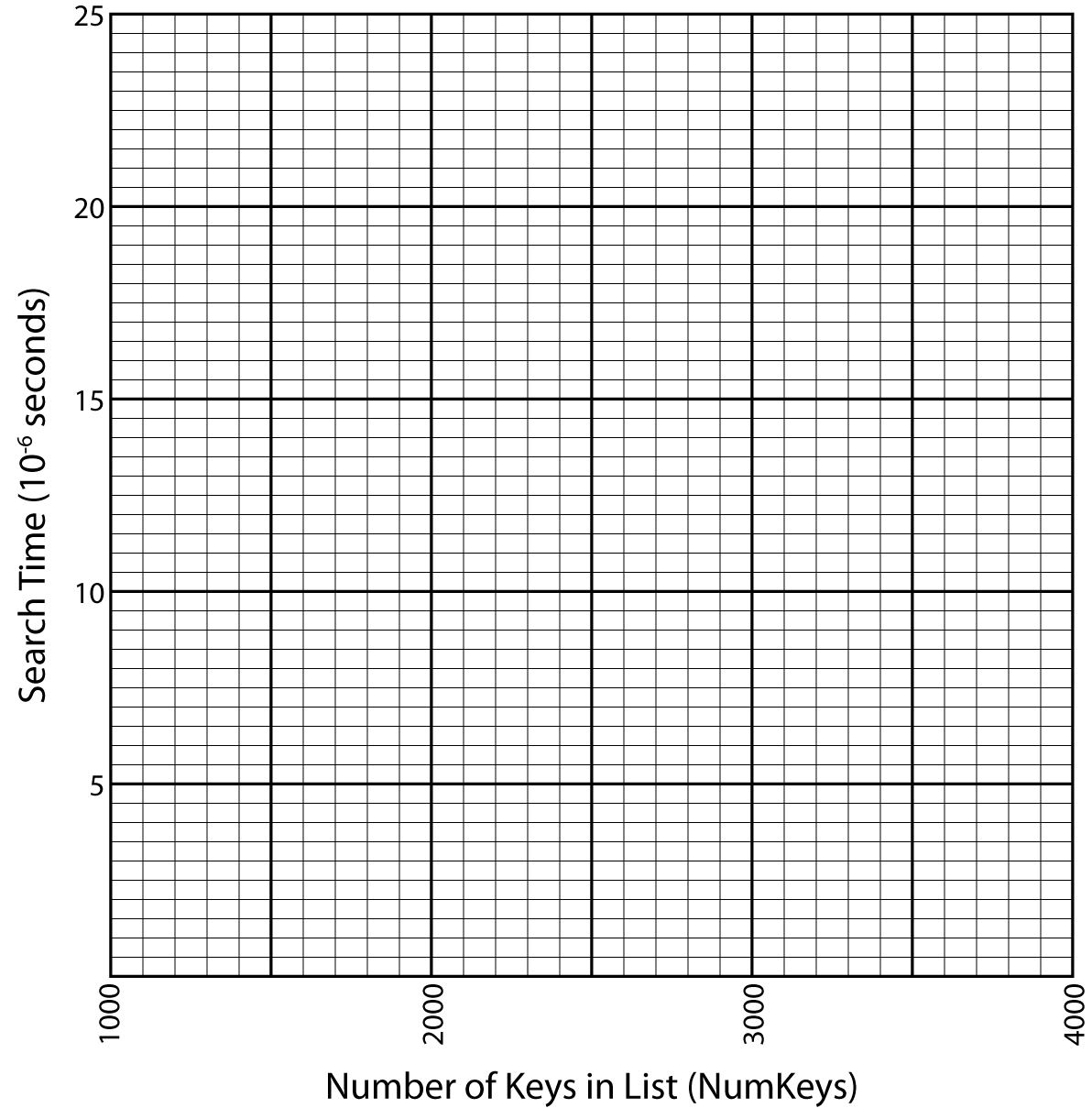
**Step 2**: Complete the following table by recording the **Time per Search** of the linearSearch(), binarySearch(), and unknownSearch() routines for each of the values of numKeys listed in the table.

**Execution Times of a Set of Searching Routines**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine | Number of Keys | | | |
| 1000 | 2000 | 3000 | 4000 |
| linearSearch() *O( N )* |  |  |  |  |
| binarySearch() *O( logN )* |  |  |  |  |
| unknownSearch() *O( )* |  |  |  |  |

***2 marks. 1*** *for “correct” values – all work done on lab computers should be very similar.* ***1*** *for correct magnitude. All answers should be converted to and recorded in ×10-6 to simplify graphing.*

**Step 3**: Plot the results below



***6 marks. 1*** *for line or point labels that identify which routine corresponds to which line.* ***1*** *for making clearly visible points.* ***1*** *for making tidy lines or curves.* ***3*** *for accurate plotting – one mark per line; I want all points to be close to and on the correct side of the nearest grid line.*

**Step 4**: Consider the two routines in the table above with known **big O** execution times. Which one should run faster? Describe the expected shape of the curve for both algorithms.

***6 marks.*** ***2*** *for which is faster (binarySearch).* ***4*** *for describing curvature of lines (linear should be straight line, binary search is curved with decreasing derivative – it is levelling off but never constant.).****Give half marks*** *if they are clearly only referring to the graph and not the* ***big O****.*

**Step 5**: Using the code in the file search.h and your measured execution times as a basis, what is the execution time of the unknownSearch() routine. Briefly explain your reasoning behind this estimate.  
***4 marks.*** *Big O (N).* ***1*** *for statement of correct big O.* ***1*** *for correct interpretation of graph.* ***1*** *for any reference to N dependent loop(s).* ***1*** *one for “good” analysis of code.*

**Part 2 – Performance of Sorting Routines**

The following questions are taken from *A laboratory Course in C++ Data Structures*, by Roberge, Brandle and Whittington:

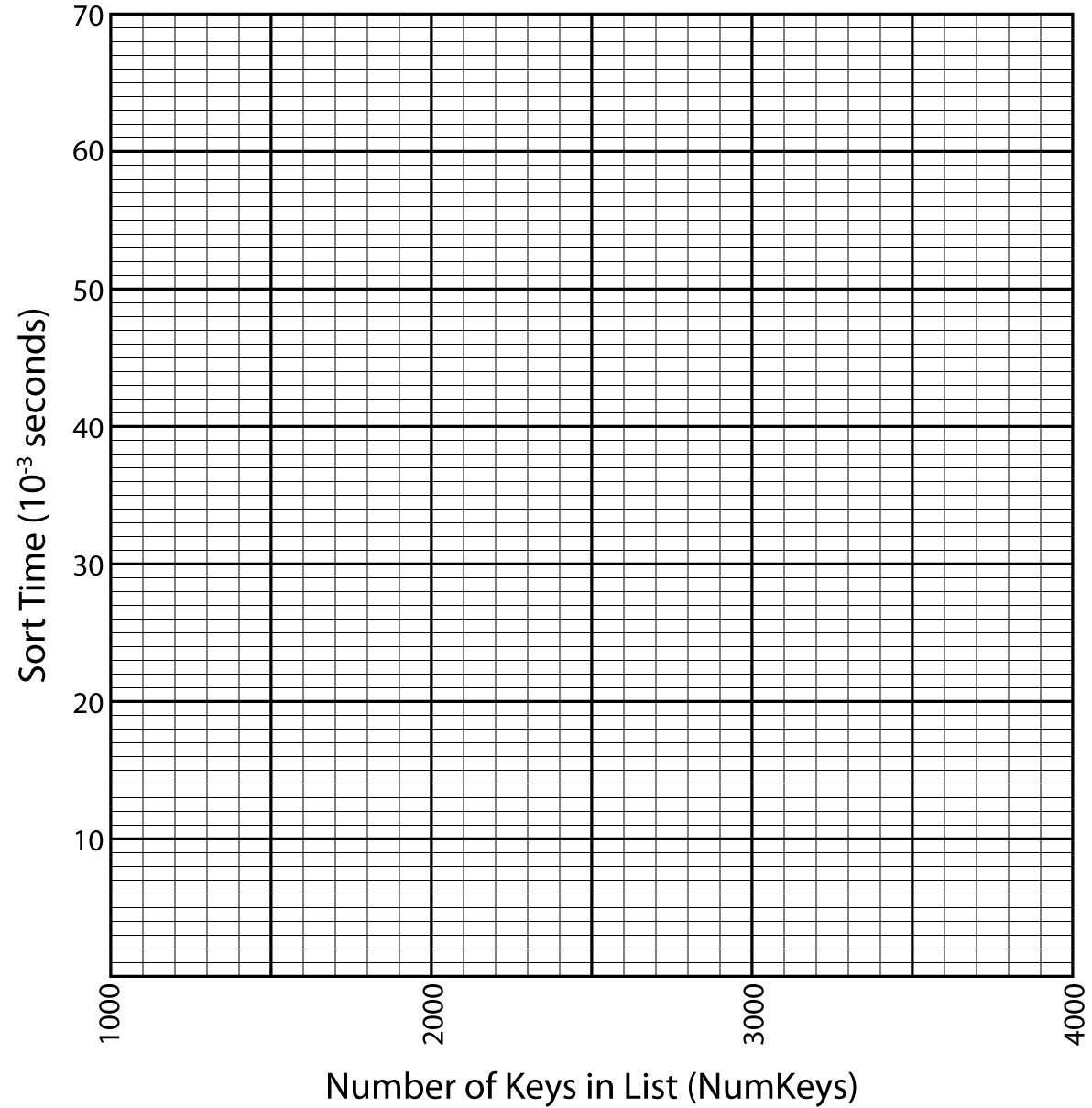
**Step 2**: Complete the following table by recording the **Time per Sort** of the selectionSort(), quickSort(), and unknownSort() routines for each of the values of numKeys listed in the table.

**Execution Times of a Set of Sorting Routines**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine | Number Of Keys | | | |
| 1000 | 2000 | 3000 | 4000 |
| selectionSort() *O( N2 )* |  |  |  |  |
| quickSort() *O( N logN )* |  |  |  |  |
| unknownSort() *O( )* |  |  |  |  |

***2 marks.*** *Same as for previous, except all answers should be in ×10-3.*

**Step 3**: Plot the results below



***6 marks.*** *Exactly like before.*

**Step 4**: Consider the two routines in the table above with known **big O** execution times. Which one should run faster? Describe the expected shape of the curve for both algorithms.

***6 marks.*** *Distribution as before.**quickSort is fastest. All lines are curved upward,but quickSort is much flatter –the curvature is slight and decreasing – approaches linear but remains super-linear.*

**Step 5**: Using the code in the file sort.h and your measured execution times as a basis, what is the execution time of the unknownSort() routine. Briefly explain your reasoning behind this estimate.  
***4 marks.*** *Big O (N2). Mark as before.*

***Teaching note:*** *Many students think this one might be Big O(N3) because it grows faster than selectionSort and they seem to have no instinct for monotonic increasing curves beyond “straight” and “not straight” . Make sure they understand constant multipliers, and show both the selectionSort and the unknownSort code during the lecture. I also point out, when I demonstrate the N2 measurement example in the notes, that doubling N will quadruple the measured time. Smart students will point this out in the measured times.*