Given a set $S$ of $n$ numbers and a target value $x$, the **membership problem** is to answer the question “Is $x$ in $S$?” There are numerous ways to answer this question – some are quite sophisticated, but in this assignment we will consider the two simplest:

**Algorithm A**: Store the set $S$ in an array or list, in whatever order the values are given. Search the list from beginning to end. If $S[i] = x$ for any value of $i$, stop searching and return “Yes”, otherwise return “No”

**Algorithm B**: Store the set $S$ in an array or indexed list (such as the Python list structure). Sort the set. Use binary search to determine if $x$ is in $S$. Return “Yes” or “No” as appropriate.

When using Algorithm A, searching for a value that is in the set will require an average of $n/2$ comparisons, while searching for a value that is not in the set will always require $n$ comparisons.

When using Algorithm B, searching for any value will require an average of about $\log n$ comparisons. However, sorting the set will take $O(n\log n)$ time.

**Now suppose we are making multiple searches.** More precisely, given a set $S$ and a sequence of values $x_1, x_2, \ldots x_k$ we want to determine whether or not each $x_i$ is in $S$. We can adapt the two algorithms as follows:

Algorithm MA:
- Store $S$ in an array or list, in whatever order it happens to be in.
- For each $x_i$:
  - Search the array from beginning to end. If $S[j] = x_i$ for any value of $j$, stop searching and print “Yes”. Otherwise, print “No”

Algorithm MA should run in $O(kn)$ time.
Algorithm MB:
   Store S in an array or indexed list.
   Sort the set using an $O(n\log n)$ sort.
   For each $n_i$:
      Use binary search to determine if $n_i$ is in S. Print “Yes” or “No” as appropriate.

Algorithm MB should run in $O((n+k)\log n)$ time

If we are doing a very small number of searches, Algorithm MA is preferable. However if we are doing many searches of the same set, Algorithm B is preferable since the time required to sort the set once is more than offset by the reduced time for the searches. At least, this is what complexity theory tells us. Your task is to conduct experiments to explore the relationship between the size of the set and the number of searches required to make Algorithm B preferable to Algorithm A.

**Part 1:**

Implement both Algorithm MA and Algorithm MB. When implementing Algorithm MB, you must write your own sort function and your own binary search function. You may use any sorting algorithm that has complexity in $O(n\log n)$. For the purposes of this assignment you may include Quick-Sort in this category.

Python, Java, C and C++ all provide clock functions. Add calls to appropriate clock functions to your implementations so that you can measure the actual running time of the two algorithms. **Make sure you include the time to complete the sorting when you time Algorithm B.**
Part 2:

For \( n = 1000, 2000, 4000, 8000 \) and 16000, conduct the following experiment:

1. Use a pseudo-random number generator to create a set \( S \) containing \( n \) integers
   (duplicate numbers in the set are ok)

2. For values of \( k \) ranging from \( 10^1 \) upwards \(^2\):
   
   Choose \( k \) target values, half of which are in \( S \) and half are not in \( S \) (see below for a simple method of doing this). Duplicate target values are ok – in fact, if you use test cases where \( k > 2^n \), they are necessary!

   Use Algorithm MA to search the set for the \( k \) target values. Note the total time required. Repeat 500 times and take the average. (500 trials is more than adequate to give strong confidence that your average value is meaningful.)

   Use Algorithm MB to search the set for the \( k \) target values. Note the total time required. Repeat 500 times and take the average.

3. Determine the approximate smallest value of \( k \) for which Algorithm MB becomes faster than Algorithm MA – note that this may be greater than \( n \). Call this value \( F(n) \).

Create a table or graph showing your values of \( F(n) \) as a function of \( n \).

If your chosen language does not have clock functions that can distinguish between Algorithm MA and Algorithm MB for the values of \( n \) indicated above, increase \( n \) until you are able to obtain meaningful results.

To easily create a set of search values, half of which are in \( S \) and half of which are not:

- when generating the set \( S \), use only even integer values
- randomly choose some elements of \( S \) as the first half of the target set
- randomly choose odd integer values as the second half of the target set

---

1 You may find that this is much too low – if so, you can start \( k \) at higher values
2 Your maximum value of \( k \) will be determined experimentally. As explained in the assignment, you need to increase \( k \) until you reach the point at which Algorithm MB is faster than Algorithm MA. This will depend on your implementation so I cannot tell you in advance what it will be.
Part 3:

Analyze your experimental results to answer the following question:

As $n$ increases, does the ratio of $F(n)/n$ increase, decrease or remain relatively constant?

Report your results and analysis in a text file or pdf file.

Logistics of Completing and Submitting Your Work

This assignment is to be completed individually.

Your solution to this assignment must be submitted by 11:59 PM on Friday, January 22. Please submit your solution using onQ.

The programming component of this assignment can be completed in Python, Java, C, or C++.

Your source code must contain your full name and student number, as well as the following statement as a comment:

“I confirm that this submission is my own work and is consistent with the Queen’s regulations on Academic Integrity.”

Your submission should consist of your source code, the tabulated results of your experiment, and your answer to the question posed in Part 3. Please combine all files into a zip archive before uploading. Your zip archive must be named using the following naming convention:  <student number>_Assignment_1.zip

Late assignments will be penalized at 10% per day up to a maximum of 3 days late (72 hours after the submission deadline). Assignments will not be accepted more than 72 hours after the submission deadline.

Before submitting your solution to this assignment, please complete the online quiz on onQ regarding academic integrity. Your assignment will only be graded if you have completed the quiz and scored 100% on it.