**Efficiency Table for Library**

Note: The most efficient time is bolded.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Insert (all methods set to default add 500 books) | Search (uses prebuilt array with five book ID’s, including the first book, the last (500th) book, and a non-existent book) | Print All | Delete (same as Search) |
| Dictionary | 00:00:00.0520466 | **00:00:00.0006705** | **00:00:00.5199772** | **00:00:00.0043991** |
| List | **00:00:00.0460257** | 00:00:00.0007658 | 00:00:00.5511612 | 00:00:00.0108107 |
| Tree | 00:00:00.0951321 | 00:00:00.0022942 | 00:00:00.6860550 | 00:00:00.0065301 |

**Summary**:

I found that the timing changed each time I ran the program, so in order to properly assess the efficiency I would recommend building a program which runs each method many times and then creates an average. I feel that these averages would better represent the accurate efficiency of each data structure. However, based on the above timing, it would seem that the dictionary is generally the most efficient.

**Note**: An attempt at calculating the average time was added to the end of the program page. While it still has inconsistencies, by calculating the average based only on a middle range of times, the average time for the actions was generally more accurate and reliable.