**Assignment: Search and sort techniques**

Searching refers to looking for a specific data point (e.g., a number) in a data structure. Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order, such as numerical (1234…) or alphabetical (abcd…) order. Search and sort algorithms differ in efficiency, so it is important to know how they work.

**Readings:**

<https://python-textbok.readthedocs.io/en/1.0/Sorting_and_Searching_Algorithms.html>

<http://interactivepython.org/courselib/static/pythonds/SortSearch/toctree.html>

**Videos:**

Binary Search -<https://youtu.be/IcK2Qyk3cUs>

Insertion Sort -<https://youtu.be/Nkw6Jg_Gi4w>

Selection Sort -<https://youtu.be/mI3KgJy_d7Y>

Bubble Sort -<https://youtu.be/YHm_4bVOe1s>

Merge Sort -<https://youtu.be/Nso25TkBsYI>

Quick Sort -<https://youtu.be/CB_NCoxzQnk>

Sorting Algorithms Benchmark Comparison -<https://youtu.be/bJ0aERNrErA>

**Assignment**

1. Use the following sorted list of words to perform a binary search:

|  |  |
| --- | --- |
| **index** | **value** |
| 0 | babka |
| 1 | baklava |
| 2 | cheesecake |
| 3 | cupcake |
| 4 | danish |
| 5 | eclair |
| 6 | funnelcake |
| 7 | kringle |
| 8 | lamington |
| 9 | profiterole |
| 10 | sopaipilla |
| 11 | strudel |
| 12 | tiramisu |
| 13 | torte |
| 14 | turnover |

1. What sequence of 'middle' values are compared to the target when performing a binary search with target *doughnut*?
2. What sequence of 'middle' values are compared to the target when performing a binary search with target *tiramisu*?

2. For the list below,

a) show the series of steps taken by the Bubble Sort algorithm while sorting this list.

b) show the series of steps taken by the Selection Sort algorithm while sorting this list.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| value | 9 | 20 | 6 | 10 | 14 | 8 | 60 | 11 |

3.The three search algorithms have varying run times, with the *unsorted sequential search* being the least efficient and the *binary search* being the most efficient (among the three).

1. Modify the code for the [seq\_search](http://cs.slu.edu/~fritts/CSCI140_F12/schedule/python/unsortedSearch.py.txt).py (for unsorted lists), [seq\_search](http://cs.slu.edu/~fritts/CSCI140_F12/schedule/python/sortedSearch.py.txt)\_ordered.py and binary\_search.py to count the number of elements checked during the search.
2. Using your instrumented code for the three search algorithms, perform the following searches, and report the number of elements checked by each algorithm for each search (i.e. you will report nine results in total -- for each of the 3 searches below, you will report the results from the 3 different search algorithms).

i. Search for the value 9 in the following array:

A = [6, 19, -3, 5, 12, 7, 21, -8, 25, 10, 0, 28, -6, 1, 33, 18, 9, 2, -13, 43]

ii. Search for the value 11 in the following array:

A = [6, 19, -3, 5, 12, 7, 21, -8, 25, 10, 0, 28, -6, 1, 33, 18, 9, 2, -13, 43]

iii. Search for the value 11 in the following array:

A = [6, 19, -3, 5, 12, 7, 21, -8, 25, 10, 0, 28, -6, 1, 33, 18, 9, 2, -13, 43, -15, 4, 22, 38, -5, 13, 23, -11, 29, -20, 41, 31, -23, 35, 40, 14, 8, -18, 16, 36]

4. Using a random number generator, create a list of 50 integers and a list of 1000 integers. Perform a benchmark analysis using **merge sort, quick sort, bubble sort and selection sort** (algorithm are given) on each of the lists. For each list, what is the difference in execution speed between the different sorting techniques?