1. Searching Algorithms

Given a data structure that stores data, one of the most common operations is to search that data structure for specific data.

For a simple data structure, such as a 1D array or an ArrayList, there are two searching algorithms that are primarily used:

Linear Search
Binary Search

Both of these algorithms were covered in CSE110 so we will very briefly review them before moving on.

1. Searching Algorithms :: Linear Search

If we have an ArrayList object and the elements of the ArrayList are in no particular order—that is they are not sorted—then we can locate a specific element K by starting at the first element of the ArrayList and checking to see if that is K. If it is, we are done. If it is not, we move to the second ArrayList element and check to see if it is K. If it is, we are done, but if if not then we continue this process to the third, fourth, fifth, ..., last element. If K is in the list, we are guaranteed to eventually stumble across it, and if it is not, we will figure that out when we reach the end of the list.

Here is the algorithm, presented in Java-like pseudocode (the item we are searching for is commonly referred to as the **key**):

```
public int linearSearch(ArrayList<T> pList, T pKey) {
  for (int i = 0; i < pList.size(); ++i) {
    if (pList.get(i) == pKey) {
      return i;
    }
  }
  return -1; // Not found
}
```

The method returns the index of *pKey* within *pList* or -1 if *pKey* is not in *pList*.

1. Searching Algorithms :: Binary Search

If we have an ArrayList object and the elements of the ArrayList are sorted in either ascending (smallest to largest) or descending (largest to smallest) order, then we can more quickly locate the key by taking advantage of the fact that the list is sorted.

```
public int binarySearch(ArrayList<T> pList, T pKey) {
  int low = 0, high = pList.size() - 1;
  while (low <= high) {
    int middle = (low + high) / 2;
    // Found pKey. Return the index which is the value of middle.
    if (pKey == pList.get(middle)) {
      return middle;
    // pKey is in the bottom half of the list. Move high down.
    } else if (pKey < pList.get(middle)) {
      high = middle - 1;
    // pKey is in the top half of the list. Move low up.
    } else {
      low = middle + 1;
    }
  }
  return -1; // Not found
}
```