Searches and Sorting

Jerry Zhu

Searches VS Sorts

- A search iterates through a list of data for the purpose of finding a target value.
- Example: Your math teacher asks you to develop an algorithm to find the highest mark on the new math test. One would develop a search to find the largest number in the array of marks.

- A sort, just like real life, moves values in a of data set based on a certain criteria.
- Example: Your math teacher asks you to develop an algorithm to sort the marks on the new test. Here, one would make a sort to arrange an array of marks from least to greatest.

Needless to say, these two are both computer algorithms dealing with the management of data.

Searching Algorithms

For the purpose of the AP Exam, one would only need to know the following 2:

- **Sequential Search** - an O(n) search that iterates through every element in the list, trying to find the target value. This search is very inefficient for large data sets as on average, it takes many checks to find the correct value is found.
- **Binary Search** - an O(log2n) search that works ONLY for sorted lists, but is one of the more efficient searches. The algorithm divides the list in half and checking the middle position. From there, it decides greater the target value is in the first or second half. It repeats to check the middle value until it finds the target.

Analysis of Searches

- Sequential search is effective for operating on small data sets because it simply iterates through every element. However...
- Never use it on large sets of data unless you want to eat all of your computer's RAM. For the same reason that they are effective on small data sets, it be very inefficient if we're talking thousands of elements because it goes through all of them individually.
- For this, binary search is to be preferred given the array is already sorted. For example, for an array of 1,000,000 elements, it only take 20 iterations max to find the target value(O(log2 n)).

Sorting Algorithms

The following 2 Sorting Algorithms are the most important for the AP Exam:

- **Selection sort -** this O(n^2) algorithm finds the smallest element in a list and placing it at the beginning repeatedly. An array being sorted with selection sort can be thought as split into two parts: the beginning, with the sorted elements, and the end, with the unsorted ones.
- **Insertion sort -** this O(n^2) algorithm passes through the array and immediately places each element into its correct position by sliding it towards the beginning until it is “inserted” into its correct place.

Analysis of the Sorts

- The aforementioned sorts are only effective on small data sets. If the list is already partially sorted, selection sort will be more efficient, but with fairly unsorted lists, insertion sort runs in less time.
- Never use either of these sorts on large sets of data. For both sorts, an enormous amount of iterations is needed.
- For this, “divide-and-conquer” algorithms such as mergesort and quicksort are optimal. However, they almost never come up on the exam so they are not covered.

General Tips

- Avoid IndexOutOfBoundsExceptions by tracing your algorithm, instead of immediately trying to alter your code. Common places to check for are the indexes of your loops and the parameters of the methods that operate the elements.
- Use the most efficient sort for each scenario to save time and space.
- If a search or sort doesn't seem to work, print out every element in the list before and after the algorithm to see what, if anything, was changed.

Personal Reflection

- This unit provided me with an understanding of algorithms on a very profound level; the content of which will follow me throughout my post-secondary education as I will be pursuing a statistics major which applies algorithms extensively.
- It gave me insight into the variety of algorithms there are and the versatility and power they have.